The Dissertation Committee for William Michael Hoza
certifies that this is the approved version of the following dissertation:

Derandomizing Space-Bounded Computation via Pseudorandom Generators and their Generalizations

Committee:

David Zuckerman, Supervisor

Pooya Hatami

Adam Klivans

Dana Moshkovitz

Derandomizing Space-Bounded Computation via Pseudorandom Generators and their Generalizations

by

William Michael Hoza

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

August 2021

Acknowledgments

Being a graduate student has been a privilege. I am enthusiastically grateful to everyone who has made this experience possible.

I thank David Zuckerman for being an excellent advisor. First and foremost, David has been kind, encouraging, and understanding. Working with him has been a pleasure. As a bonus, David's technical comments are consistently wise, insightful, and illuminating. He has been an invaluable collaborator, and his optimistic attitude about solving difficult problems is inspiring.

I thank the rest of the UT CS faculty – especially Scott Aaronson, Anna Gal, Adam Klivans, Dana Moshkovitz, Eric Price, and Brent Waters – for their roles in creating a pleasant and welcoming environment, especially at the beginning of my graduate studies when I was exploring different research directions and I had not yet chosen an advisor. Scott and Adam in particular provided much indispensable guidance.

I thank all the researchers with whom I have had the pleasure of coauthoring papers during graduate school: Kuan Cheng, Dean Doron, Cole Graham, Pooya Hatami, Adam Klivans, Ted Pyne, Avishay Tal, Roei Tell, Salil Vadhan, and David Zuckerman. I appreciate each of these coauthors for teaching me about the theory of computation, the research process, the art of writing, and more. I also thank the countless students and researchers from whom I learned in the context of courses, seminars, conferences, casual conversations,

and collaborations that did not quite lead to publishable papers.

I thank those who got me interested in the theory of computing prior to graduate school, especially Leonard Schulman and Chris Umans. Through instruction, mentoring, and collaboration, they gave me an excellent introduction to this wonderful field.

I thank the National Science Foundation and the University of Texas Graduate School for their generous fellowships that have supported me throughout graduate school. (My graduate studies have been supported by the NSF GRFP under Grant DGE-1610403 and by a Harrington Fellowship.) In turn, I thank U.S. taxpayers for supporting agencies such as the NSF. Of course, I understand that taxpayer support is not entirely voluntary; I hope that it is not resented. I also thank everyone who made it financially possible for me to travel to conferences, workshops, etc. during graduate school through various types of grants.

I thank my wife Alicia for her companionship, support, and love. I have especially appreciated her friendship during this past year while the COVID-19 pandemic has impeded socialization. I also thank her (partially in advance) for graciously tolerating certain aspects of academic life such as relocating for jobs and traveling for conferences.

Last but not least, I thank my creator for setting up this whole grand universe with my seemingly-inconsequential graduate studies in mind.

William Michael Hoza

The University of Texas at Austin
August 2021

Derandomizing Space-Bounded Computation via Pseudorandom Generators and their Generalizations

William Michael Hoza, Ph.D.
The University of Texas at Austin, 2021

Supervisor: David Zuckerman

To what extent is randomness necessary for efficient computation? We study the problem of deterministically simulating randomized algorithms with low space overhead. The traditional approach to this problem is to try to design sufficiently powerful *pseudorandom generators* (PRGs). PRG constructions often provide deep insights into models of computation and have applications beyond derandomization. There are other approaches based on *generalizations* of the PRG concept that are potentially easier to construct yet still valuable for derandomization. One such generalization is a *hitting set generator* (HSG), a standard "one-sided" version of a PRG. Another such generalization, introduced recently by Braverman, Cohen, and Garg [BCG20], is a *weighted pseudorandom generator* (WPRG), which is similar to a PRG except probability distributions are replaced by "pseudodistributions." In this dissertation, we present new results (obtained jointly with collaborators) regarding all

three of these approaches to derandomizing space-bounded algorithms.

Regarding the PRG approach, we present improved PRGs for interesting models of computation including unbounded-width permutation branching programs, read-once $\mathbf{AC}^0$ formulas, and superlinear-size constant-depth threshold circuits. The first two PRGs are unconditionally near-optimal; substantially improving the third would require a breakthrough in circuit lower bounds. Each PRG is in some way connected to derandomizing space-bounded algorithms.

Regarding generalizations of PRGs, we present new constructions and applications of HSGs and WPRGs for *read-once branching programs* (ROBPs), the nonuniform model that directly corresponds to randomized space-bounded algorithms. In particular, we construct explicit HSGs and WPRGs with an optimal dependence on the error parameter ε. In terms of applications, we show that optimal HSGs for polynomial-width ROBPs could be used to derandomize decision algorithms with two-sided error (**BPL**, not just **RL**). Unconditionally, we explain how to use recent WPRG constructions to deterministically simulate randomized space-S decision algorithms in space $O\left(S^{3/2}/\sqrt{\log S}\right)$, a slight improvement over Saks and Zhou's celebrated $O(S^{3/2})$ bound [SZ99]. Finally, using the techniques underlying our HSG construction, we give an improved unconditional derandomization of log-space algorithms that have one-sided error and low success probability.

Table of Contents

List of Tables

Chapter 1

Introduction

1.1 Derandomization

Randomization is a hugely successful technique in algorithm design. In practice, randomized algorithms are often faster or more space-efficient than their deterministic counterparts. However, randomness can itself be considered a computational resource in limited supply, just like time and space. We want to use as few random bits as possible, just like we want to conserve other types of algorithmic "fuel." We should therefore ask, have we had more success with randomized algorithms because of *inherent* limitations of deterministic computation? Or are we merely experiencing the limitations of human algorithm designers?

In some settings, such as communication complexity, randomized algorithms provably have an advantage. In the setting of conventional decision algorithms, however, the popular conjecture is that randomness is not necessary for efficient computation (assuming somewhat coarse standards of efficiency). Concretely, if a decision problem can be solved in time $T \geq N$ on inputs of length N[1] using randomness, then conjecturally it can be solved deterministically

[1]Throughout this dissertation, we denote a uniform algorithm's input length by uppercase N rather than the traditional lowercase n. That way, n is available for denoting the number of random bits used by the

in time poly(T). If it can be solved in space $S \geq \log N$ using randomness, then conjecturally it can be solved deterministically in space $O(S)$.

Recall that **P** and **L** are the classes of languages that can be decided by deterministic algorithms running in polynomial time and logarithmic space respectively. **BPP** and **BPL** are defined analogously, but we allow bounded-error randomized algorithms.[2] By elementary arguments,

$$\mathbf{L} \subseteq \mathbf{BPL} \subseteq \mathbf{P} \subseteq \mathbf{BPP}.$$

The conjectures in the previous paragraph are essentially equivalent to the conjectures $\mathbf{P} = \mathbf{BPP}$ and $\mathbf{L} = \mathbf{BPL}$.

The most basic reason to believe $\mathbf{P} = \mathbf{BPP}$ and $\mathbf{L} = \mathbf{BPL}$ is simply that it seems counterintuitive for randomness to be useful for computation. The fact that randomization is genuinely helpful for solving communication problems (for example) can be considered paradoxical, and we should not let it mislead us regarding **BPP** and **BPL**. To make a common analogy, solving a problem in **NP** or **NL** is like finding a needle in a haystack, whereas solving a problem in **BPP** or **BPL** is almost like finding *hay* in a haystack! It is strange to think that it might be intrinsically hard.

However, after decades of research, nobody has managed to actually prove $\mathbf{P} = \mathbf{BPP}$ or $\mathbf{L} = \mathbf{BPL}$. There is widespread pessimism regarding the goal of unconditionally derandomizing **BPP**. Proving $\mathbf{P} = \mathbf{BPP}$ would require proving serious new circuit lower bounds [KI04; AM11; JS12; CIKK15], which realistically will not happen anytime soon. The good news is that there is much more hope regarding **L** vs. **BPL** (the main topic of this dissertation). To prove $\mathbf{L} = \mathbf{BPL}$, it does not seem to be necessary to prove new lower bounds. The human species has no real excuse for having not already proven $\mathbf{L} = \mathbf{BPL}$, and if we are lucky we will prove $\mathbf{L} = \mathbf{BPL}$ in the near future.

---

algorithm (or the number of pseudorandom bits we are trying to generate, etc.)

[2] There are some subtleties in the definition of **BPL**; see Section 2.2.

Derandomizing **BPL** could *conceivably* have a direct practical impact. Modern computing often requires processing datasets that are so enormous that they cannot feasibly be stored in working memory, and hardware random number generators can only output so much entropy per second. One can *imagine* efficient, general derandomization methods, with rigorous mathematical proofs of correctness, being implemented on actual computers.

More importantly (and more realistically), **L** vs. **BPL** is a fundamental scientific question. By studying the algorithmic power of randomness, we can elucidate the true nature of computation. From the beginning, one of the central goals of computational complexity theory has been to clarify the relationships between different computational resources, and that includes space and randomness.

1.2 Pseudorandom Generators (PRGs)

A powerful approach for proving **L** = **BPL** (or **P** = **BPP**) is to design *pseudorandom generators* (PRGs). Let U_n denote the uniform distribution over $\{0,1\}^n$.

Definition 1.2.1. *Let $n \in \mathbb{N}$, let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^n \to \mathbb{R}$, let X be a distribution over $\{0,1\}^n$, and let $\varepsilon > 0$. We say that X* fools *$\mathcal{F}$* with error *ε, or* ε-fools *$\mathcal{F}$, if for every $f \in \mathcal{F}$,*

$$|\,\mathbb{E}[f(X)] - \mathbb{E}[f(U_n)]| \leq \varepsilon.$$

An ε-PRG *for $\mathcal{F}$ is a function $G\colon \{0,1\}^s \to \{0,1\}^n$ such that $G(U_s)$ fools $\mathcal{F}$ with error ε. We also speak of G itself fooling $\mathcal{F}$.*

Intuitively, a PRG stretches a short truly random "seed" $y \in \{0,1\}^s$ out to a long pseudorandom string $G(y) \in \{0,1\}^n$. It is desirable to have a short seed length and a small error. To see the connection to derandomization, let $A(x,z)$ be a randomized decision algorithm, where x is the input and z is the random bits. Define $A_x(z) = A(x,z)$. Suppose

that for each input x, the function A_x is in the class $\mathcal{F}$ fooled by G. Then we can define a new randomized algorithm $A'(x, y) = A(x, G(y))$. For each input x, we have

$$\Pr[A'(x) = 1] \approx \Pr[A(x) = 1],$$

but A' uses fewer random bits than A (assuming G has a nontrivial seed length).

If A is a randomized polynomial-time algorithm (corresponding to **BPP**), then A_x can be computed by a *polynomial-size circuit*, so it would be great to have PRGs for circuits. Meanwhile, if A is a randomized log-space algorithm (corresponding to **BPL**), then A_x can be computed by a polynomial-width *read-once branching program*[3] (ROBP), as we will explain in Section 2.4. An ROBP is a generalization of a finite automaton where we allow a different transition function in each step.

Definition 1.2.2. *Let $w, n \in \mathbb{N}$. A width-w length-n* ROBP *is a directed graph. The vertices are arranged in $n + 1$ layers with w vertices per layer. Each vertex has two outgoing edges labeled* 0 *and* 1 *leading to the next layer, except the vertices in the last layer which have no outgoing edges. The program processes an input $x \in \{0, 1\}^n$ by starting at a designated* start vertex v_{start} *in the first layer, reading the bits of x from left to right to decide which edges to traverse, and accepting or rejecting depending on whether it arrives at a designated* accept vertex v_{acc} *in the final layer. Thus, the program computes a function $f\colon \{0, 1\}^n \to \{0, 1\}$. We often identify the program with the function it computes.*

PRGs for width-n length-n ROBPs can be used to decrease the randomness used by log-space decision algorithms. In either the polynomial-time or the log-space setting, to

[3]Note that Definition 1.2.2 requires the program to read the input bits in order, from left to right. By using the phrase "read-once branching program" to refer to this model, we are following the standard conventions of the pseudorandomness literature, despite the fact that the phrase does not clearly express the ordering assumption. Indeed, outside the context of pseudorandomness, the term "read-once branching program" typically refers to a more general model of computation that does not require ordering or even obliviousness.

eliminate randomness altogether, we can try all seeds exhaustively and take a majority vote, i.e., define a deterministic algorithm A'' by

$$A''(x) = \mathsf{MAJ}(A(x, G(0^s)), A(x, G(0^{s-1}1)), \ldots, A(x, G(1^s))).$$

Clearly, if A has low failure probability, A'' is correct. The efficiency of this derandomization depends on the seed length of the PRG and the efficiency of the PRG. To derandomize **BPP** or **BPL**, we would like a PRG with seed length $O(\log n)$ that can be computed in $\mathrm{poly}(n)$ time or $O(\log n)$ space, respectively. Next, we discuss reasons to believe that such PRGs exist. In turn, the likely existence of such PRGs is arguably the best reason to believe $\mathbf{P} = \mathbf{BPP}$ and $\mathbf{L} = \mathbf{BPL}$.

1.2.1 Reasons to Believe That Good PRGs Exist

Nonexplicit PRGs via the probabilistic method

If we set aside the issue of efficiently computing the PRG, then there is a well-known, generic, probabilistic argument that shows that PRGs exist with a good seed length.

Proposition 1.2.3. *Let $n \in \mathbb{N}$, and let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^n \to \{0,1\}$ with $|\mathcal{F}| \geq 2$. For every $\varepsilon > 0$, there exists a PRG for $\mathcal{F}$ with seed length $\log\log|\mathcal{F}| + 2\log(1/\varepsilon) + O(1)$.*

Proof. Sample a generator G uniformly at random from the set of all functions $G\colon \{0,1\}^s \to \{0,1\}^n$. Then for each $f \in \mathcal{F}$, the value $2^{-s} \cdot \sum_x f(G(x))$ is an average of 2^s Bernoulli random variables, each with expectation $\mathbb{E}[f(U_n)]$. By Hoeffding's inequality, it is in the interval $\mathbb{E}[f(U_n)] \pm \varepsilon$ except with failure probability $2e^{-2\varepsilon^2 \cdot 2^s}$. Therefore, by the union bound, G is an ε-PRG for $\mathcal{F}$, except with failure probability $2|\mathcal{F}|e^{-2\varepsilon^2 \cdot 2^s}$. For $s = \log\log|\mathcal{F}| + 2\log(1/\varepsilon) + O(1)$, this failure probability is less than 1, so a suitable G exists. □

Corollary 1.2.4. *There exist PRGs for size-s circuits on n input bits and width-w length-n ROBPs with seed lengths $O(\log(s/\varepsilon) + \log\log n)$ and $O(\log(wn/\varepsilon))$ respectively.*

Corollary 1.2.4 shows that good PRGs exist as functions; the challenge is to come up with a genuine *algorithm* that efficiently computes such a PRG. The proof of Proposition 1.2.3 can be considered "evidence" (albeit weak evidence) that such algorithms exist. After all, by default, we expect that probabilistic arguments can be matched by explicit constructions. Relatedly, one can readily devise plausible candidate PRG constructions [LV17]. The hardest part is proving that the constructions work.

Hardness and conditional PRGs

Another source of evidence for the existence of good PRGs is a long line of work obtaining *conditional* constructions of PRGs under unproven complexity-theoretic or cryptographic assumptions [Yao82; BM84; NW94; IW97; HILL99; KM02; Uma03; DMOZ20; CT20]. For derandomizing **BPP**, Impagliazzo and Wigderson were the first to prove the following striking and celebrated theorem [IW97].

Theorem 1.2.5 ([IW97])**.** *Assume there is a language that can be decided deterministically in time $2^{O(N)}$ and that has circuit complexity $2^{\Omega(N)}$, where N is the input length. Then there is a 0.1-PRG for size-n circuits on n input bits with seed length $O(\log n)$ that is computable in time $\mathrm{poly}(n)$, and consequently $\mathbf{P} = \mathbf{BPP}$.*

Klivans and van Melkebeek showed a similar conditional derandomization of **BPL** [KM02].

Theorem 1.2.6 ([KM02])**.** *Assume there is a language that can be decided deterministically in space $O(N)$ and that has branching program complexity $2^{\Omega(N)}$, where N is the input length.*

Then there is a 0.1*-PRG for width-*n *length-*n *ROBPs*[4] *with seed length* $O(\log n)$ *that is computable in space* $O(\log n)$*, and consequently* $\mathbf{L} = \mathbf{BPL}$.

The "hardness" assumptions in Theorems 1.2.5 and 1.2.6 seem likely to be true, so these theorems give us fairly compelling reasons to believe that good explicit PRGs exist. The connection between PRGs and hardness also goes "the other way." Intuitively, a PRG construction is a strong kind of impossibility result for whatever model of computation the PRG fools, because the PRG gives an example of a task that cannot be solved by the model, namely distinguishing the output of the PRG from the uniform distribution. More concretely, a PRG also implies a hard decision problem. Starting from a PRG $G\colon \{0,1\}^s \to \{0,1\}^n$, we can define $f_G\colon \{0,1\}^{s+1} \to \{0,1\}$ by

$$f_G(x) = 1 \iff \exists y,\ G(y)_{1\ldots s+1} = x.$$

Clearly, $\mathbb{E}[f_G(U_{s+1})] \leq 1/2$, but $\mathbb{E}[f_G(G(U_s)_{1\ldots s+1})] = 1$, so the truncation of G to the first $s+1$ bits does not fool f_G. This means that f_G is hard relative to the class of functions that G is guaranteed to fool.

The upshot is that PRGs can give deep insights regarding the abilities and limitations of the model. Unfortunately, understanding some models of computation seems to be far beyond us, so this perspective on PRGs can be interpreted as a *barrier* to actually constructing good PRGs (even if they exist). For example, if we could construct a PRG G for size-n circuits with seed length $O(\log n)$ that could be computed in $\mathrm{poly}(n)$ time, we would get a function $f_G\colon \{0,1\}^\ell \to \{0,1\}$ that could be computed in $2^{O(\ell)}$ time with circuit complexity $2^{\Omega(\ell)}$. This exactly matches the assumption of Theorem 1.2.5, and proving the existence of such an f_G would be a huge breakthrough. Realistically, the takeaway is that there is little

[4] In fact, this conditional PRG fools all polynomial-size branching programs, including read-many programs. Such a PRG implies the stronger derandomization result $\mathbf{L} = \mathbf{BP}^*\mathbf{L}$ (see Section 2.2.1).

hope of constructing such a PRG G unconditionally in the foreseeable future. This is bad news for the project of derandomizing **BPP**.

What about **L** vs. **BPL**? Is there a similar barrier? If we could construct an explicit PRG G for width-n length-n ROBPs with seed length $O(\log n)$, we would get a function $f_G\colon \{0,1\}^\ell \to \{0,1\}$ such that any ROBP computing f_G has width $2^{\Omega(\ell)}$. But that is not any barrier at all, because it is actually easy to *unconditionally* construct hard functions for ROBPs. For example, consider the "inner product mod 2" function $\mathsf{IP}\colon \{0,1\}^{\ell/2} \times \{0,1\}^{\ell/2} \to \{0,1\}$ defined by $\mathsf{IP}(x,y) = \bigoplus_{i=1}^{\ell/2}(x_i \cdot y_i)$. If a width-$w$ ROBP can compute IP, then we get a two-party communication protocol for IP: Alice simulates the first half of the ROBP's computation and sends Bob $\lceil \log w \rceil$ bits specifying the state reached in the middle layer, then Bob simulates the second half of the ROBP's computation. By standard communication complexity lower bounds, it follows that $w \geq 2^{\Omega(\ell)}$, i.e., IP is hard for ROBPs.

This is good news for derandomizing **BPL**. To construct PRGs for ROBPs, the challenge is "merely" to bridge the gap between hardness and pseudorandomness. For this reason, it makes sense to be much more optimistic about proving **L** = **BPL** than about proving **P** = **BPP**.

PRGs for weak models and/or with weak parameters

The next reason to believe that there exist PRGs powerful enough to prove **L** = **BPL** is that we *already* have explicit PRGs, unconditionally, that are quite interesting and powerful, though they fall short of fully derandomizing **BPL**. For width-n length-n ROBPs, Babai, Nisan, and Szegedy constructed the first explicit PRG, with seed length $2^{O(\sqrt{\log n})} \cdot \log(1/\varepsilon)$ [BNS92]. Nisan gave a significant improvement in a subsequent game-changing paper [Nis92]:

Theorem 1.2.7 ([Nis92])**.** *For every $w, n \in \mathbb{N}$ and every $\varepsilon > 0$, there exists an explicit*[5]

[5]We will say a generator $G\colon \{0,1\}^s \to \{0,1\}^n$ is *explicit* if it runs in space $O(s)$. Of course, this only really makes sense if there is an infinite family of generators, indexed by one or more parameters (in this

ε-PRG for width-w length-n ROBPs with seed length $O(\log(wn/\varepsilon) \cdot \log n)$.

The seed length of Nisan's PRG is only a $\log n$ factor away from optimal. Nisan's PRG is a crucial ingredient in Nisan's later proof that **BPL** is contained in **SC** [Nis94] and Saks and Zhou's later proof that **BPL** is contained in $\mathbf{DSPACE}(\log^{3/2} n)$ [SZ99]. Furthermore, Nisan's PRG and other PRGs for ROBPs [INW94; NZ96] have found applications beyond the derandomization of space-bounded decision algorithms [Siv02; Ind06; HVV06; HHR11; CL21].

Nisan's PRG remains the best explicit PRG known for width-n length-n ROBPs to this day. However, explicit optimal PRGs are known for more restricted models of computation, such as *parity functions*. A distribution over $\{0,1\}^n$ is called *ε-biased* if it ε-fools all functions of the form $\bigoplus_{i \in I} x_i$ where $I \subseteq [n]$. An *ε-biased generator* is a PRG whose output is ε-biased.

Theorem 1.2.8 ([NN93; Per90; AGHP92])**.** *For every $n \in \mathbb{N}$ and every $\varepsilon > 0$, there exists an explicit ε-biased generator with seed length $O(\log(n/\varepsilon))$.*

Parity functions are a weak class on their own, but optimal or near-optimal PRG constructions are known for more interesting models of computation, many of which use Theorem 1.2.8 as a building block. We will see an example in Section 5.2.

Nisan's PRG (Theorem 1.2.7) demonstrates that it is possible to design explicit PRGs for ROBPs with a highly nontrivial seed length. Meanwhile, theorems such as Theorem 1.2.8 show that it is possible to design explicit PRGs for *some* models with optimal seed length. Taken together, these results suggest that we should expect to be able to remove the extra $\log n$ factor from the seed length of Nisan's generator (at least in the absence of any real argument for why the extra $\log n$ factor would be necessary). This gives even more evidence that $\mathbf{L} = \mathbf{BPL}$.

case, w, n, and ε). The algorithm for computing $G(x)$ is given the parameters and the seed x.

1.2.2 Our PRG Contributions

As mentioned, Nisan's generator [Nis92] is still the best explicit PRG known for width-n length-n ROBPs, three decades later. However, there has been a great deal of success designing improved PRGs for *related* models of computation. Some highlights include PRGs for "regular" and "permutation" ROBPs [BV10; De11; KNP11; Ste12; RSV13; BRRY14; CHHL19; HPV21], PRGs for "arbitrary-order" ROBPs [BPW11; SVW17; CHRT18; FK18], PRGs for combinatorial rectangles and their variants [ASWZ96; EGLNV98; Lu02; GMRTV12; Wat13; GMRZ13; Vio14; De15; GKM18; HLV18; Lee19; GY20], PRGs for width-3 ROBPs [SVW17; MRT19], and PRGs for multiple-read branching programs [INW94; DPS11; IMZ19; GV20; HHTT21]. The hope is that eventually, we will develop enough PRG-related skills and knowledge that we can design a better PRG for width-n length-n ROBPs.

In Chapter 5, we continue in this tradition by presenting new unconditional PRGs for several interesting models of computation. Each PRG is connected in some way to space-bounded derandomization. We briefly list the PRGs here, deferring to Chapter 5 for more detailed discussion of context, prior work, and techniques.

PRGs for unbounded-width permutation ROBPs A *permutation* ROBP is an ROBP such that no two edges with the same label point to the same vertex. In Section 5.1, we prove that an explicit PRG (the INW generator [INW94]) fools *unbounded-width* permutation ROBPs (with a single accept vertex) with seed length $\widetilde{O}(\log n \cdot \log(1/\varepsilon))$.[6] Conversely, we prove that any ε-PRG for these programs must have seed length at least $\widetilde{\Omega}(\log n \cdot \log(1/\varepsilon))$. These results are joint work with Ted Pyne and Salil Vadhan [HPV21].

PRGs for read-once $\mathbf{AC}^0$ In Section 5.2, we present an explicit PRG for read-once $\mathbf{AC}^0$ formulas with near-optimal seed length $\widetilde{O}(\log(n/\varepsilon))$. This generator is joint work with Dean

[6] Throughout this dissertation, we use the notation $\widetilde{O}(s) = s \cdot \operatorname{polylog} s$ and $\widetilde{\Omega}(s) = s/\operatorname{polylog} s$.

Doron and Pooya Hatami [DHH19].

PRGs for slightly-superlinear-size $\mathbf{TC}^0$ In Section 5.3, we present an explicit PRG for depth-d threshold circuits with $n^{1+\delta}$ wires with seed length $O(n^{1-\delta})$ for some $\delta = 2^{-O(d)}$. The PRG is joint work with Pooya Hatami, Avishay Tal, and Roei Tell [HHTT21]. Our seed length is only slightly nontrivial, but significant improvements would require new lower bounds for $\mathbf{TC}^0$.

1.3 Generalizations of the PRG Concept

Since designing PRGs has turned out to be very challenging, researchers have considered relaxations of the PRG concept. The idea is that these generalized PRGs are potentially easier to construct, but they still have value for derandomization.

1.3.1 Hitting Set Generators (HSGs)

The first "generalized PRG" we will consider is a *hitting set generator* (HSG).

Definition 1.3.1. *Let $n \in \mathbb{N}$, let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^n \to \{0,1\}$, let $H \subseteq \{0,1\}^n$, and let $\varepsilon > 0$. We say that H is an* ε-hitting set *for $\mathcal{F}$ if for every $f \in \mathcal{F}$,*

$$\mathbb{E}[f(U_n)] \geq \varepsilon \implies \exists x \in H, f(x) = 1.$$

An ε-HSG *for $\mathcal{F}$ is a function $G\colon \{0,1\}^s \to \{0,1\}^n$ such that the image $G(\{0,1\}^s)$ is an ε-hitting set for $\mathcal{F}$.*

Clearly, if G is an ε-PRG for $\mathcal{F}$, then it is also an ε'-HSG for $\mathcal{F}$ for any $\varepsilon' > \varepsilon$. Now, suppose we wish to derandomize an algorithm A that decides a language L. Using an HSG

G, we can try to derandomize A by computing

$$\bigvee_{y \in \{0,1\}^s} A(x, G(y)).$$

Recall that $A_x(z) \stackrel{\text{def}}{=} A(x, z)$. The derandomization is correct as long as G is an ε-HSG for the A_x functions, A succeeds with probability at least ε, and A has *one-sided error*, i.e.,

$$\begin{aligned} x \in L &\implies \Pr[A(x) = 1] \geq \varepsilon \\ x \notin L &\implies \Pr[A(x) = 1] = 0. \end{aligned}$$

This condition corresponds to complexity classes such as **RL** and **RP**, which are contained in **BPL** and **BPP** respectively. It is quite common for randomized algorithms to have one-sided error, so it is immediately clear that HSGs have value for derandomization. HSGs have been studied since the 80s [AKS87][7] and are considered standard today.

1.3.2 Weighted Pseudorandom Generators (WPRGs)

The other relaxation of PRGs we study is a *weighted pseudorandom generator* (WPRG), also known as a pseudorandom pseudodistribution generator.

Definition 1.3.2. *Let $n \in \mathbb{N}$, let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^n \to \mathbb{R}$, and let $\varepsilon > 0$. An* ε-WPRG *for $\mathcal{F}$ is a pair (G, ρ), where $G\colon \{0,1\}^s \to \{0,1\}^n$ and $\rho\colon \{0,1\}^s \to \mathbb{R}$, such that for every $f \in \mathcal{F}$,*

$$\left| \mathop{\mathbb{E}}_{x \sim U_s} [\rho(x) \cdot f(G(x))] - \mathbb{E}[f(U_n)] \right| \leq \varepsilon.$$

[7] In the same paper [AKS87], Ajtai, Komlos, and Szemeredi conjectured that (in today's terminology) **L** = **RL**. They were perhaps the first to do so. Another historical note: Aleliunas, Karp, Lipton, Lovasz, and Rackoff seem to have been the first to raise the *question* of whether **L** = **RL** [AKLLR79].

We also say that (G, ρ) fools $\mathcal{F}$ *with error* ε. *If* ρ *maps* $\{0,1\}^s \to [-K, K]$, *we say that* (G, ρ) *is* K-bounded.

WPRGs were introduced relatively recently by Braverman, Cohen, and Garg [BCG20]. WPRGs can be viewed as a generalization of PRGs, because if G is an ε-PRG for $\mathcal{F}$, then $(G, 1)$ is an ε-WPRG for $\mathcal{F}$. Meanwhile, a WPRG can be viewed as an HSG with extra structure, because if (G, ρ) is an ε-WPRG for $\mathcal{F}$, then G is an ε'-HSG for $\mathcal{F}$ for every $\varepsilon' > \varepsilon$. (After all, if $\mathbb{E}[f(U_n)] > \varepsilon$, then $\mathbb{E}_{x \sim U_s}[\rho(x) \cdot f(G(x))] > 0$, so there must be some seed x such that $f(G(x)) \neq 0$.) Thus, we have a hierarchy,

$$\text{PRG} \implies \text{WPRG} \implies \text{HSG}.$$

If A is a randomized decision algorithm, we can try to derandomize it using a WPRG by computing the weighted sum $\sum_{y \in \{0,1\}^s} 2^{-s} \cdot \rho(y) \cdot A(x, G(y))$ and comparing to $1/2$. This derandomization will work as long as (G, ρ) fools the A_x functions with sufficiently low error, even if A has two-sided error. For example, optimal WPRGs for ROBPs would immediately imply $\mathbf{L} = \mathbf{BPL}$.

Recall that Nisan's PRG ε-fools width-n length-n ROBPs with seed length

$$O(\log^2 n + \log n \cdot \log(1/\varepsilon)).$$

In the paper introducing the WPRG concept [BCG20], Braverman, Cohen, and Garg presented an explicit WPRG that ε-fools width-n length-n ROBPs with seed length

$$\widetilde{O}(\log^2 n + \log(1/\varepsilon)),$$

which is better than Nisan's PRG when $\varepsilon \ll 1/\operatorname{poly}(n)$. This was especially exciting because prior to Braverman, Cohen, and Garg's work, it was not even known how to construct an

HSG for width-n length-n ROBPs with a better seed length than Nisan's generator (except when ε is exponentially small; see Table 1.1). Their work prompted a flurry of activity investigating WPRGs and the small-ε regime [HZ20; CL20; CDRSTS21; PV21; Hoz21b], some of which we will discuss in this dissertation.

1.3.3 Our Contributions Regarding HSGs and WPRGs

In Chapters 3 and 4, we present new constructions and applications of HSGs and WPRGs for ROBPs. Once again, we briefly describe the results here but defer to Chapters 3 and 4 for more detailed discussion.

Generators for the small-ε regime In Sections 3.1 and 4.1 respectively, we present an ε-HSG and an ε-WPRG for width-n length-n ROBPs with seed length

$$O(\log^2 n + \log(1/\varepsilon)).$$

The HSG is joint work with David Zuckerman [HZ20]. These constructions remove $\log\log$ factors from Braverman, Cohen, and Garg's result [BCG20] and subsequent improvements by Chattopadhyay and Liao [CL20], Cohen, Doron, Renard, Sberlo, and Ta-Shma [CDRSTS21], and Pyne and Vadhan [PV21] (see Table 1.1). The WPRG is more general, but we include the HSG construction as well, because it is especially simple and because it gives a better seed length for short, wide ROBPs. Both constructions work via modular *reductions* that can convert any moderate-error PRG for ROBPs into a low-threshold HSG or a low-error WPRG.

A conditional application of HSGs In Section 3.3, we prove that optimal HSGs for width-n length-n ROBPs would imply $\mathbf{L} = \mathbf{BPL}$, not just $\mathbf{L} = \mathbf{RL}$. The analogous theorem

Seed length	Type of generator	Reference
$\widetilde{O}(\sqrt{n}) + O(\log(1/\varepsilon))$	HSG	[AKS87]
$2^{O(\sqrt{\log n})} \cdot \log(1/\varepsilon)$	PRG	[BNS92]
$O(\log^2 n + \log(1/\varepsilon) \cdot \log n)$	PRG	[Nis92]
$\widetilde{O}(\log^2 n + \log(1/\varepsilon))$	WPRG	[BCG20]
$O(\log^2 n + \log(1/\varepsilon))$	HSG	[HZ20] (Section 3.1)
$\widetilde{O}(\log^2 n) + O(\log(1/\varepsilon))$	WPRG	[CL20]
$O(\log^2 n) + \widetilde{O}(\log(1/\varepsilon))$	WPRG	[CDRSTS21; PV21]
$O(\log^2 n + \log(1/\varepsilon))$	WPRG	[Hoz21b] (Section 4.1)

Table 1.1: Known explicit constructions of PRGs, WPRGs, and HSGs for width-n length-n ROBPs, in chronological order. Ajtai, Komlos, and Szemeredi were focused on ROBPs with length much smaller than their width [AKS87], but their work also immediately implies a nontrivial HSG for the width-n length-n case as listed here. The papers of Cohen, Doron, Renard, Sberlo, and Ta-Shma [CDRSTS21] and Pyne and Vadhan [PV21] are independent of each other and simultaneous with each other.

for **BPP** was already known; the **BPL** case requires different techniques (see Section 3.3.1). This result is joint work with Kuan Cheng [CH20].

Improved unconditional derandomization In Section 4.2, we point out that using the recent work on WPRGs, it is possible to unconditionally improve the state-of-the-art derandomization of space-bounded computation. As mentioned previously, Saks and Zhou showed how to deterministically simulate a randomized space-S decision algorithm in space $O(S^{3/2})$ [SZ99]; we demonstrate the improved bound $O\left(S^{3/2}/\sqrt{\log S}\right)$. (This result does not require our improved WPRG construction.) Furthermore, in Section 3.2, we present an improved unconditional derandomization of **RL** in the *small-success* regime, i.e., algorithms with one-sided error that accept positive instances with some small probability $\varepsilon > 0$. The latter result is joint work with David Zuckerman [HZ20] and uses the same techniques as our HSG construction.

1.4 Discussion: PRGs vs. WPRGs vs. HSGs

One of the goals of this dissertation is to clarify the relative virtues of PRGs, WPRGs, and HSGs. The basic tradeoff is of course that HSGs are easier to construct whereas PRGs are more useful; WPRGs are intermediate in both respects. We now elaborate.

Difficulty of constructing unweighted PRGs

As discussed, we now have WPRGs and HSGs for width-n length-n ROBPs that are better than the best known unweighted PRGs (see Table 1.1). There are many other cases where good HSGs have been constructed for some model of computation and no matching PRG construction is known [AKS87; LLSZ97; RS10; BDS13; GMRTV12; ŠŽ11; Lu12; BRRY14; DTST20; CHMY21] or at least none was known at the time [ACR99; ISW99; MV05; SU05].

These facts already demonstrate that on some level, it is easier to construct WPRGs and HSGs than it is to construct unweighted PRGs.

In the setting of unbounded-width permutation ROBPs with a single accept vertex (see Section 5.1), unweighted PRGs are actually *provably* harder to construct, in the following sense. We will prove that any $(1/n)$-PRG for this model must have seed length $\Omega(\log^2 n)$ (Theorem 5.1.19). In contrast, in joint work with Pyne and Vadhan, the author showed by a probabilistic argument that there exists a $(1/n)$-HSG for this model with seed length $O(\log n)$ [HPV21]. Furthermore, Pyne and Vadhan subsequently gave an explicit construction of a $(1/n)$-WPRG for this model with seed length $\widetilde{O}(\log^{3/2} n)$ [PV21]. Thus, in at least one case, there is an *inherent gap* between PRGs on the one hand and HSGs and WPRGs on the other.

Relative usefulness

As mentioned previously, we will show that explicit optimal HSGs for ROBPs would imply $\mathbf{L} = \mathbf{BPL}$, so they would be essentially just as useful as explicit optimal WPRGs. However, when it comes to non-optimal generators (what we have in real life), WPRGs are more useful, because they can be plugged into the Saks-Zhou framework [SZ99], provided they are K-bounded for a sufficiently small K. This fact was first demonstrated by Chattopadhyay and Liao [CL20], who argued that WPRGs with certain parameters could hypothetically be used to deterministically simulate randomized space-S decision algorithms in space $O(S^{4/3})$ (developing an earlier suggestion by Braverman, Cohen, and Garg [BCG20]). We will see in Section 4.2 that WPRGs can be used *today* to achieve the weaker bound $O(S^{3/2}/\sqrt{\log S})$. It remains an open problem to devise a way to combine HSGs with the Saks-Zhou methodology.

Meanwhile, unweighted PRGs for ROBPs are more useful than WPRGs or HSGs because unweighted PRGs can more easily be used as building blocks in *randomized* algorithms. For example, PRGs for ROBPs are commonly used in the design of randomized

streaming algorithms (a technique introduced by Indyk [Ind06]), and it is not clear how HSGs or WPRGs could play the same role. For another example, PRGs are often key building blocks for constructing *more PRGs*. If we could construct an explicit unweighted PRG for ROBPs matching the current WPRG constructions, we could use it to build an explicit PRG for constant-width ROBPs with seed length $O(\log^{3/2} n)$ [CH20, Lemma 4.9], which would be a huge breakthrough.

HSGs and simplicity

Work on HSGs is often simpler and easier to understand than work on PRGs. Constructions and analyses of PRGs typically involve analytical insights that could be described as "deep," whereas constructions and analyses of HSGs sometimes only use combinatorial tricks that are better described as "clever." One striking example is the work of Braverman, Rao, Raz, and Yehudayoff on regular ROBPs [BRRY14]. Braverman et al. use randomness extractors and information-theoretic arguments to construct a PRG with seed length $\widetilde{O}(\log n)$ for constant-width regular ROBPs. Then, by a short combinatorial proof, they show that simply taking all low-Hamming-weight strings gives an optimal hitting set for the same model.

Our results on HSGs fit this pattern as well. Our construction of HSGs for ROBPs (Section 3.1) and our proof that optimal HSGs derandomize **BPL** (Section 3.3) are both fairly elementary, making them approachable for those with relatively little background in the area. Indeed, our work on HSGs has been included in at least two courses so far, one taught by Amnon Ta-Shma [TS18] and another taught by Chris Umans [Uma20]. For this reason, we present our work on HSGs first (in Chapter 3) before moving on to WPRGs (Chapter 4) and unweighted PRGs (Chapter 5).

1.5 Omitted Work

Some of the author's work is omitted from this dissertation, because it is not sufficiently relevant, because the research was done prior to the author starting graduate school, or because we deem it to be of lesser importance. We briefly mention a few results that almost made the cut; the curious reader is invited to read the relevant papers for details.

In addition to the PRGs that we will discuss in this dissertation, the author has constructed improved PRGs for certain classes of read-once $\mathbf{AC}^0[\oplus]$ formulas (joint with Dean Doron and Pooya Hatami [DHH19; DHH20]), as well as for De Morgan formulas and unrestricted branching programs (joint with Pooya Hatami, Avishay Tal, and Roei Tell [HHTT21]). Meanwhile, using classic PRGs for space-bounded computation [Nis92; NZ96], the author has proven a *typically-correct* log-space derandomization of quasilinear-time log-space decision algorithms [Hoz19b]. Finally, in joint work with Kuan Cheng, the author has shown that if there are explicit optimal HSGs for constant-width ROBPs, then it is possible to estimate $\mathbb{E}[f] \pm \varepsilon$ in space $O(\log(n/\varepsilon))$ given only black-box access to a constant-width ROBP f [CH20].

Chapter 2

Preliminaries

2.1 Deterministic Space-Bounded Computation

As usual, our model of deterministic space-bounded computation is a Turing machine with a read-only input tape, a read-write work tape, and a write-only output tape. We say that the machine runs in space $S = S(N)$ if it touches at most S cells of the work tape on inputs of length N, which is a perfectly natural definition provided $S(N) \geq \log N$ so the algorithm has enough space to store a pointer into the input. (There are some interesting results regarding the derandomization of sublogarithmic-space Turing machines [Fre81; MS82; Tom81; Jun84; KV87; Mac98], but sublogarithmic space complexity is sensitive to the specific model of computation, so we will not study such algorithms.) We let $\mathbf{DSPACE}(S)$ denote the class of languages that can be decided by a deterministic algorithm that runs in space $O(S)$, and $\mathbf{L} = \mathbf{DSPACE}(\log n)$.

It is well-known that space-bounded algorithms compose nicely. For example, if $F\colon \{0,1\}^* \to \{0,1\}^*$ is length-preserving and can be computed in space S, then the i-fold composition of F with itself can be computed in space $O(i \cdot S)$. The following lemma

is a generalization that will be useful for deterministically simulating randomized space-S decision algorithms in space $o(S^{3/2})$ (Section 4.2). The idea is that we consider a function with two inputs, $F(x, y)$. The first input x comes from a recursive computation, so reading x is "expensive," whereas the second input y does not participate in the recursion, so reading y is "cheap."

Lemma 2.1.1. *Let $F\colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$. Suppose that for all $y \in \{0,1\}^*$, the function $F(\cdot, y)$ is length-preserving. Suppose also that there is a deterministic algorithm that computes $F(x, y)$ given $x \in \{0,1\}^{N_1}$ and $y \in \{0,1\}^{N_2}$ in space $S = S(N_1, N_2)$, and each time it reads a bit from x, it first clears all but $T = T(N_1, N_2)$ bits from its work space. Define*

$$F^{(0)}(x) = x$$
$$F^{(i+1)}(x, y_1, \ldots, y_{i+1}) = F(F^{(i)}(x, y_1, \ldots, y_i), y_{i+1}).$$

Then given $i \geq 1$, $x \in \{0,1\}^{N_1}$, and $y_1, \ldots, y_i \in \{0,1\}^{N_2}$, the value $F^{(i)}(x, y_1, \ldots, y_i)$ can be computed deterministically in space $O(S + i \cdot (T + \log(N_1 N_2)))$.

Proof sketch. To compute $F^{(i)}(x, y_1, \ldots, y_i)$, we run the algorithm for $F(\cdot, y_i)$. Each time it tries to read a bit from its first input, we pause its computation and recursively (re)compute $F^{(i-1)}(x, y_1, \ldots, y_{i-1})$ to obtain that bit. While the outer computation is paused, the only data regarding that outer computation that we need to store is the T bits on its work space, plus $\log(N_1 N_2)$ bits indicating the positions of its read heads. At any given time, at most one of the recursive simulations will be "active" and using up to S bits of processing space. □

2.2 Randomized Space-Bounded Computation

Intuitively, **BPL** is the randomized counterpart to **L**, just like **BPP** is the randomized counterpart to **P**. However, there are some subtleties in modeling randomized space-bounded computation. We give a more precise definition below.

Definition 2.2.1. *Let $S = S(N)$ be a function. We define* **BPSPACE**(S) *to be the class of languages that admit a Turing machine A with the following properties.*

1. *There is a read-only input tape, a read-write work tape, and a read-only "random tape" filled with uniform random bits.*

2. *For every $N \in \mathbb{N}$, on every input $x \in \{0,1\}^N$, the machine touches at most $O(S(N))$ cells of the work tape, and*

$$x \in L \implies \Pr[A(x) = 1] \geq 2/3 \tag{2.1}$$

$$x \notin L \implies \Pr[A(x) = 1] \leq 1/3. \tag{2.2}$$

3. *The machine only has* one-way *access to the random tape, i.e., the read head for the random tape can move right but not left.*

4. *The machine* always halts*, i.e., it halts for every input and every setting of the random tape.*

We define **BPL** $=$ **BPSPACE**$(\log n)$*. We define* **RSPACE**(S) *and* **RL** *similarly, except that Eqs. (2.1) and (2.2) are replaced by*

$$x \in L \implies \Pr[A(x) = 1] \geq 1/2$$

$$x \notin L \implies \Pr[A(x) = 1] = 0.$$

The last two conditions in the definition of **BPSPACE** are the less obvious ones, especially for those who are more familiar with **BPTIME**. We discuss them in turn.

2.2.1 The One-Way Random Tape Condition

We only allow one-way access to the random tape, because we are trying to model *computing with access to a fair coin* rather than computing with access to a database of random bits. If an algorithm is wondering about the outcome of a coin flip that happened earlier in the computation, the algorithm should have written it down at the time and paid for it in terms of space complexity.

Still, it is natural to be curious about $\mathbf{BP^*L}$, the variant of **BPL** where we allow random access to the random bits. Klivans and van Melkebeek's conditional proof of $\mathbf{L} = \mathbf{BPL}$ (Theorem 1.2.6) actually establishes $\mathbf{L} = \mathbf{BP^*L}$ under the same assumption. Therefore, it seems likely that $\mathbf{L} = \mathbf{BPL} = \mathbf{BP^*L}$ and there is no inherent gap between **BPL** and $\mathbf{BP^*L}$.

In terms of what we can actually prove, however, there is a huge gap. We are not even able to prove $\mathbf{BP^*L} \neq \mathbf{PSPACE}$. We could derandomize $\mathbf{BP^*L}$ given an explicit PRG for unrestricted (read-many) branching programs, but designing such a PRG with a short seed would require a breakthrough in lower bounds. For this reason, intuitively, the $\mathbf{L}$ vs. $\mathbf{BP^*L}$ problem feels more similar to $\mathbf{P}$ vs. **BPP** than it does to $\mathbf{L}$ vs. **BPL**. In fact, there is a formal connection: $\mathbf{L} = \mathbf{BP^*L}$ implies a nontrivial derandomization of **BPTIME** [MPV15]. Note also that **BPL** is contained in $\mathbf{ZP^*L}$ (the zero-error version of $\mathbf{BP^*L}$) [Nis93a], which is another reason to think that derandomizing **BPL** is easier than derandomizing $\mathbf{BP^*L}$.

2.2.2 The Requirement of Always Halting

In our definition of $\mathbf{BPSPACE}(S)$, we require that the algorithm always halts. Without this requirement, randomized space-bounded algorithms are surprisingly powerful. For ex-

ample, consider the following log-space algorithm [Gil77] for the **NL**-complete problem of s-t connectivity in an n-vertex 2-outregular directed graph.

1. Take a random walk of length n from s. If vertex t is reached, halt and accept.
2. Make $100n$ coin tosses. If all of them come up "heads," halt and reject.
3. Go back to step 1.

If there is a path from s to t, then a random walk from s of length n will visit t with probability at least 2^{-n}, so the algorithm will accept with high probability. Conversely, if the algorithm accepts, then obviously there is a path from s to t. The algorithm does not necessarily ever halt – but it does halt with probability 1!

Non-halting algorithms like the one above are quite interesting, but we do not allow them in our definition of **BPL**. One reason for this decision is that we are trying to model *efficient* randomized computation, and non-halting algorithms such as the one above might have exponential expected running time. In contrast, if a randomized log-space algorithm always halts, then it necessarily does so in polynomial time:

Proposition 2.2.2. *Let A be a randomized log-space algorithm that always halts, like in Definition 2.2.1. Then for every $N \in \mathbb{N}$, on every input of length N and every setting of the random tape, A halts within* $\mathrm{poly}(N)$ *steps.*

Proof. Assume without loss of generality that A reads a random bit in every step. Fix some N-bit input x. We can specify the contents of the work tape, the location of the read head, and the internal state of the machine using $c \log N$ bits for some constant c. Let $w = N^c$, and define a w-vertex directed graph G, where each non-halting configuration u has two outgoing edges indicating which configuration the machine will enter next depending on the random bit it reads in that step. Let s be the vertex corresponding to the initial configuration of A,

and assume for a contradiction that there is some vertex u that is reachable from s that is part of a cycle. Let y be the sequence of edge labels on the path from s to u, and let z be the sequence of edge labels on the cycle from u back to u. Then $A(x, yzzz\cdots)$ never halts, a contradiction. Therefore, G has no cycles reachable from s, and therefore A always halts within w steps. □

(Conversely, if a language can be decided by a bounded-error randomized log-space algorithm that runs in *expected* polynomial time but does not always halt, then the language is in **BPL**, i.e., it can also be decided by a bounded-error randomized log-space algorithm that *always* halts in polynomial time.)

Confusingly, in the older literature, terms like "**BPSPACE**," "**BPL**," etc. refer to the *non-halting* versions of these classes, and a different term such as "$\mathbf{BP_HL}$" or "**BPLP**" is used to refer to the class that we denote "**BPL**" today. Due to diminishing interest in the non-halting classes, no replacement name for them has become standard. This is unfortunate, especially because it seems fairly likely that the halting and non-halting versions of **BPL** do not coincide (recall that we argued that the non-halting version contains **NL**). In terms of upper bounds, the non-halting version of **BPL** is contained in uniform $\mathbf{NC}^2$ [BCP83], hence it is also contained in $\mathbf{DSPACE}(\log^2 n)$ and **P**.

For more about non-halting randomized space-bounded algorithms, we direct the reader to the surveys of Michel [Mic92] and Saks [Sak96]. As a final note, what if we give our randomized log-space machine *both* of the superpowers we have discussed – two-way access to the random tape and permission to sometimes run forever? It turns out that such machines can compute every language in **PSPACE** [KV85]! By the space hierarchy theorem [SHL65], it follows that they, at least, cannot be derandomized in small space. For the remainder of this dissertation, we return to the standard setting of algorithms with one-way access to their random tape that always halt, as stipulated in Definition 2.2.1.

2.3 Strings and Boolean Functions

We use $x \# y$ to denote the concatenation of the bitstrings x and y. For a string $x \in \{0,1\}^n$, let $x_{i\ldots j}$ denote the substring starting at position i and ending at position j. For a set $I \subseteq [n]$, let $x_I \in \{0,1\}^I$ be the string obtained by looking only at coordinates in I.

For any Boolean predicate ϕ, we let $\mathbf{1}[\phi]$ be 1 if ϕ is true and 0 if ϕ is false. If f is a function on $\{0,1\}^n$, we identify f with the random variable $f(U_n)$, so for example we write $\mathbb{E}[f]$ as a shorthand for $\mathbb{E}[f(U_n)]$. If $y \in \{0,1\}^n$, we define $f^{+y}(x) = f(x+y)$, where $+$ denotes bitwise XOR. We say that f is *ε-close to a constant* if there is some b such that $\Pr[f(U_n) = b] \geq 1 - \varepsilon$.

Let $n \in \mathbb{N}$ and let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^n \to \{0,1\}$. We say that $\mathcal{F}$ is *closed under shifts* if for every $f \in \mathcal{F}$ and every $y \in \{0,1\}^n$, we have $f^{+y} \in \mathcal{F}$. We say that $\mathcal{F}$ is *closed under complementation* if for every $f \in \mathcal{F}$, the function $1 - f$ is also in $\mathcal{F}$.

2.4 ROBPs and their Connection to BPL

The connection between ROBPs (see Definition 1.2.2) and **BPL** is spelled out in the following proposition.

Proposition 2.4.1. *Let A be a randomized log-space decision algorithm, like in Definition 2.2.1. Suppose that on inputs of length N, A uses at most $n = n(N)$ random bits, where $n \leq \mathrm{poly}(N)$. Then for every $N \in \mathbb{N}$ and every $x \in \{0,1\}^N$, there is a width-$\mathrm{poly}(N)$ length-n ROBP f_x such that for every $y \in \{0,1\}^n$, $A(x,y) = f_x(y)$. Furthermore, the ROBP f_x can be constructed in $O(\log N)$ space given x, assuming n can be computed in $O(\log N)$ space.*

Proof. The construction is similar to the graph in the proof of Proposition 2.2.2. We can specify the contents of the work tape, the location of the read head, and the internal state

of the algorithm using $c \log N$ bits for some constant c. Set $w = N^c$, and identify a number $i \in [w]$ with a configuration of the algorithm. To compute a transition of the ROBP from state $i \in [w]$ upon reading bit $b \in \{0, 1\}$, we simulate the algorithm from state i until it reads another random bit, at which time we feed it the random bit b and observe the configuration $j \in [w]$ that it moves to. We put an edge in the ROBP from state i in each layer to state j in the next layer, labeled b. (If the simulation halts before ever reading another random bit, then we put an edge from state i in each layer to state i in the next layer.) In the final layer, we merge all the accepting configurations into a single accept vertex. □

Since the algorithms defining **BPL** run in poly(N) time (Proposition 2.2.2), they use at most poly(N) random bits, hence they correspond to ROBPs of width poly(N) and length poly(N). That is why we are especially interested in ROBPs where the width and length are equal. Note that the one-way random tape condition in the definition of **BPL** (see Section 2.2.1) is the reason we get *read-once* branching programs.

In the proof of Proposition 2.4.1, the ROBP has the same transition function at every layer – it might as well have been a finite automaton. However, we would not gain anything significant by studying finite automata instead of general ROBPs, because a randomized log-space algorithm can easily estimate the acceptance probability of any given ROBP. To put it another way, define **prBPL** to be the "promise" version of **BPL**, i.e., the class of promise problems that admit randomized log-space algorithms as in Definition 2.2.1. Then the problem of determining whether a given ROBP f satisfies $\mathbb{E}[f] \leq 1/3$ or $\mathbb{E}[f] \geq 2/3$, promised that one of the two holds, is complete for **prBPL** under deterministic log-space reductions.

One benefit of looking at ROBPs is that the problem of fooling *constant-width* ROBPs is more interesting than the problem of fooling constant-size finite automata. For example, constant-width ROBPs can simulate various types of read-once formulas after possibly per-

muting the variables (see Section 5.2). For another example, one can show[1] that PRGs for constant-width ROBPs allow one to sample points in $[0,1]^D$ that fool D-dimensional geometric rectangles with respect to the uniform distribution on $[0,1]^D$.

Notation for ROBPs

We now fix some notation for ROBPs that we will use throughout the dissertation. Let f be a width-w length-n ROBP, as defined in Definition 1.2.2. We will use V to denote the vertex set of f, and we will use $V_0, \dots, V_n$ to denote the layers of f, so $V = V_0 \cup \cdots \cup V_n$. We define Wrap_f to be the 2-outregular directed graph on nw vertices obtained by identifying the i-th vertex in V_n with the i-th vertex in V_0.

If $u \in V_i$ and $T \subseteq V_{i+m}$ with $m \geq 0$, we define $f_{u \to T}$ to be a width-w length-m ROBP obtained from f by making u the start vertex and merging all vertices in T into a single accept vertex. This program computes a function $f_{u \to T} \colon \{0,1\}^m \to \{0,1\}$. For convenience, we extend $f_{u \to T}$ to a function $\{0,1\}^{\geq m} \to \{0,1\}$, where $f_{u \to T}(x)$ ignores all but the first m bits of x. Furthermore, if $m < 0$, we define $f_{u \to T} \colon \{0,1\}^* \to \{0,1\}$ to be the constant zero function.

If $v \in V_{i+m}$, we write $f_{u \to v}$ as a shorthand for $f_{u \to \{v\}}$. When the ROBP f is clear from context, we define $p_{u \to T} = \mathbb{E}[f_{u \to T}]$. We write $f_{\to T}$ and $p_{\to T}$ as a shorthand for the case $u = v_{\mathrm{start}}$, and we write $f_{u \to}$ and $p_{u \to}$ as a shorthand for the case $T = \{v_{\mathrm{acc}}\}$.

Thinking of each layer V_i as an *ordered* set of w vertices, we let $\mathrm{next}_i \colon [w] \times \{0,1\} \to [w]$ be the transition function describing the edges between V_{i-1} and V_i. That is, if the u-th vertex in V_{i-1} has an outgoing edge labeled b that leads to the v-th vertex in V_i, then $\mathrm{next}_i(u,b) = v$.

[1] David Zuckerman, personal communication.

2.5 State-of-the-Art PRGs for ROBPs

We briefly summarize the current state of the art regarding PRGs for width-w length-n ROBPs. When $w = n$, as mentioned previously, the best PRG is still Nisan's PRG (Theorem 1.2.7), with seed length $O(\log(wn/\varepsilon)\log n)$. However, better PRGs are known both when $w \ll n$ and when $n \ll w$.

For width-2 ROBPs, explicit PRGs are known with optimal seed length $O(\log(n/\varepsilon))$ [SZ95; BDVY13]. For width-3 ROBPs, the best seed length is $\widetilde{O}(\log n \cdot \log(1/\varepsilon))$ [MRT19]. For width 4, Nisan's PRG is still the best PRG.

On the other end of the spectrum, for the case $n = \operatorname{polylog} w$, Nisan and Zuckerman gave an explicit PRG with optimal seed length $O(\log w)$ and (non-optimal) error $2^{-\log^{1-\alpha} w}$, where $\alpha > 0$ is an arbitrarily small constant [NZ96]. Later, Armoni [Arm98] gave a PRG construction that interpolates between the Nisan-Zuckerman PRG [NZ96] and Nisan's PRG [Nis92]. Kane, Nelson, and Woodruff subsequently optimized Armoni's construction by plugging in a randomness extractor that was not available to him at the time [KNW08, Theorem A.16], giving the following theorem.

Theorem 2.5.1 ([Arm98; KNW08])**.** *Let $w, n \in \mathbb{N}$ and let $\varepsilon > 0$. There is an explicit ε-PRG for width-w length-n ROBPs with seed length*

$$O\left(\frac{\log(wn/\varepsilon)\log n}{\max\{1, \log\log w - \log\log(n/\varepsilon)\}}\right).$$

When $n \ll w$ and $\varepsilon = 1/\operatorname{poly}(n)$, Armoni's generator is better than Nisan's generator by a factor of $\Theta(\log\log w)$. We will use this result for our HSG construction (Section 3.1) and for our deterministic simulation of randomized space-S algorithms in space $o(S^{3/2})$ (Section 4.2). However, note that when $\varepsilon = 1/\operatorname{poly}(w)$, the savings are completely wiped out: Armoni's generator is no better than Nisan's.

Chapter 3

HSG Constructions and Applications

In this chapter,[1] we discuss the HSG approach to derandomizing space-bounded algorithms. In Section 3.1, we give a construction of an HSG for ROBPs with an optimal dependence on the threshold parameter ε. In Section 3.2, we give an unconditional derandomization of ε-success **RL** with an improved dependence on ε. Finally, in Section 3.3, we prove that optimal HSGs for ROBPs would imply $\mathbf{L} = \mathbf{BPL}$.

The first two results are joint work with David Zuckerman [HZ20], and a video of David presenting those results is available online [Zuc18]. The last result is joint work with Kuan Cheng [CH20], and there is a video presentation of it by the author available online [Hoz20].

[1] For clarity, we provide here the full citations for the papers that this chapter is based on.

- [HZ20] William M. Hoza and David Zuckerman. "Simple Optimal Hitting Sets for Small-Success RL". in: *SIAM Journal on Computing* 49.4 (2020), pp. 811–820. DOI: 10.1137/19M1268707
- [CH20] Kuan Cheng and William M. Hoza. "Hitting Sets Give Two-Sided Derandomization of Small Space". In: *Proceedings of the 35th Computational Complexity Conference (CCC)*. 2020, 10:1–10:25. ISBN: 978-3-95977-156-6. DOI: 10.4230/LIPIcs.CCC.2020.10

Each paper is a collaborative effort; in each case, each author contributed both to the intellectual development of the ideas in the paper and to the writing process.

3.1 Hitting Sets for Small-Success RL

In this section, we present a relatively simple construction of an HSG for width-w length-n ROBPs with an optimal dependence on the threshold parameter ε. This HSG is joint work with David Zuckerman [HZ20].

Theorem 3.1.1 (Joint with David Zuckerman [HZ20])**.** *For every $w, n \in \mathbb{N}$, for every $\varepsilon > 0$, there is an explicit ε-HSG for width-w length-n ROBPs with seed length*

$$O\left(\frac{\log(wn)\log n}{\max\{1, \log\log w - \log\log n\}} + \log(1/\varepsilon)\right).$$

Historically, our HSG construction came shortly after Braverman, Cohen, and Garg's WPRG for width-w length-n ROBPs [BCG20]. As a reminder, they achieved seed length

$$\widetilde{O}(\log(wn)\log n + \log(1/\varepsilon)),$$

compared to Nisan's PRG's seed length of $O(\log(wn/\varepsilon)\log n)$ (see Table 1.1). Braverman, Cohen, and Garg introduced many innovative new concepts in their intricate work. The unfortunate side effect, as they put it, is that "our construction is fairly involved and the analysis requires a significant amount of work" [BCG20].

Our seed length eliminates $\log\log$ factors from Braverman, Cohen, and Garg's seed length [BCG20]. More importantly, our construction is much simpler. We lose some of the power of their approach – this is reflected by the fact that we merely get an HSG instead of a WPRG – but we gain a significant amount of clarity. Subsequent to our work, simpler WPRG constructions have been discovered [CL20; CDRSTS21; PV21; Hoz21b], including the one that we will present in Section 4.1, but in our opinion, these WPRG constructions are still not as simple as our HSG.

When $n \geq w$, our HSG's seed length is $O(\log(wn)\log n + \log(1/\varepsilon))$. When $n \ll w$,

our seed length is slightly better. For example, consider the case $n = \operatorname{polylog} w$. Long before our work, Ajtai, Komlos, and Szemeredi gave a $(1/w)$-HSG with optimal seed length $O(\log w)$ for the case $n = O(\log^2 w / \log\log w)$ [AKS87]. In an incomparable result, Nisan and Zuckerman gave a PRG for any $n = \operatorname{polylog} w$ with optimal seed length $O(\log w)$ and non-optimal error $2^{-\log^{1-\Omega(1)} w}$. Reingold posed the problem of constructing a generator with seed length $O(\log w)$ for the case $n = \operatorname{polylog} w$ and $\varepsilon = 1/w$ [Rei10, Slide 10]. Our HSG solves this problem: when $n = \operatorname{polylog} w$, our HSG has optimal seed length $O(\log(w/\varepsilon))$. It remains an open problem to design a PRG or even a WPRG with these parameters.

Theorem 3.1.1 is proven by a generic *reduction* that converts a $(1/\operatorname{poly}(n))$-PRG for width-$w$ length-n ROBPs with seed length s into an ε-HSG with seed length $O(s + \log(wn/\varepsilon))$, for any $\varepsilon > 0$. We achieve the claimed seed length by plugging in Armoni's PRG [Arm98; KNW08].

3.1.1 Structural Lemma: Amplifying the Expectation of an ROBP

The reduction proving Theorem 3.1.1 is based on a key structural lemma regarding ROBPs. Intuitively, the lemma identifies a way to amplify the acceptance probability of the ROBP by a factor of $\lambda > 1$.

Lemma 3.1.2. *Let $\lambda > 1$, and let f be a width-w length-n ROBP with $\mathbb{E}[f] = \varepsilon \leq 1/\lambda$. There is some vertex v such that $p_{v\to} \geq \lambda\varepsilon$ and $p_{\to v} \geq \frac{1}{2wn\lambda}$.*

Proof. Let V be the set of vertices of f, and let $S = \{v \in V : p_{v\to} \in [\lambda\varepsilon, 2\lambda\varepsilon]\}$. Every accepting path must pass through S, because for any edge (u, v) in the path, u has outdegree 2, and hence $p_{u\to} \geq \frac{1}{2} p_{v\to}$, i.e., the acceptance probability at most doubles in each step along

the path. Therefore,

$$\begin{aligned}\varepsilon = \Pr_{x\sim U_n}\left[f(x)=1 \wedge \bigvee_{v\in S} f_{\to v}(x)=1\right] &\le \sum_{v\in S} p_{\to v}\cdot p_{v\to} && \text{(Union bound)}\\ &\le 2\lambda\varepsilon\cdot\sum_{v\in S} p_{\to v}\\ &\le 2\lambda\varepsilon\cdot|S|\cdot\max_{v\in S} p_{\to v}.\end{aligned}$$

Therefore, there is some $v\in S$ such that $p_{\to v}\ge \frac{\varepsilon}{2\lambda\varepsilon|S|}\ge\frac{1}{2\lambda wn}$. □

Roughly speaking, for our HSG, we wish to find a path from v_{start} to v_{acc}. We will apply Lemma 3.1.2 several times to decompose this difficult task into many easier tasks. First we find a path from v_{start} to a vertex v given by the lemma, then we find a path from v to another vertex v' given by applying the lemma again, etc. Each of these "hops" is relatively easy to perform, because the lemma guarantees that a random walk has a noticeable chance of reaching the vertex v. With each hop, we make significant progress toward reaching v_{acc}, because the acceptance probability goes up by a factor of λ.

To get the best parameters, we would like to eliminate the lemma's dependence on the width w of the branching program. To obtain this improvement, we generalize by allowing a whole *set* of target vertices v.

Lemma 3.1.3. *Let $\lambda>1$, and let f be a length-n ROBP with $\mathbb{E}[f]=\varepsilon<1/\lambda$ with layers $V_0,V_1,\dots,V_n$. There is some $i\in[n]$ and some $T\subseteq V_i$ such that $p_{\to T}\ge\frac{1}{2n\lambda}$ and for every $v\in T$, $p_{v\to}\ge\lambda\varepsilon$.*

Proof. Let $S_i=\{v\in V_i : p_{v\to}\in[\lambda\varepsilon,2\lambda\varepsilon]\}$ and $S=\cup_i S_i$. As explained in the proof of

Lemma 3.1.2, every accepting path must pass through S. Therefore,

$$\begin{aligned}
\varepsilon = \Pr_{x\sim U_n}\left[f(x)=1 \wedge \bigvee_{v\in S} f_{\to v}(x)\right] &\le \sum_{i=1}^{n} \Pr_{x\sim U_n}\left[f(x)=1 \wedge \bigvee_{v\in S_i} f_{\to v}(x)\right] \quad \text{(Union bound)} \\
&= \sum_{i=1}^{n}\sum_{v\in S_i} p_{\to v}\cdot p_{v\to} \\
&\le \sum_{i=1}^{n} p_{\to S_i}\cdot \max_{v\in S_i} p_{v\to} \\
&\le 2\lambda\varepsilon \cdot \sum_{i=1}^{n} p_{\to S_i} \\
&\le 2\lambda\varepsilon n \cdot \max_{i\in[n]} p_{\to S_i}.
\end{aligned}$$

Therefore, there is some $i \in [n]$ such that $p_{\to S_i} \ge \frac{\varepsilon}{2\lambda\varepsilon n} = \frac{1}{2n\lambda}$. Let $T = S_i$. □

Both Lemma 3.1.2 and Lemma 3.1.3 rely on the fact that each vertex has outdegree 2. For a lemma in a similar spirit regarding ROBPs over a large alphabet, see work by Cheng and the author [CH20, Lemma 4.5].

3.1.2 The Reduction: Converting $(\frac{1}{\mathrm{poly}(n)})$-PRGs into ε-HSGs

Our reduction relies on a *hitter* (equivalent to the concept of a *disperser*). See Goldreich's work [Gol11, Appendix C] for a discussion of hitters.

Definition 3.1.4. *A (β,γ)-*hitter *is a function $\mathsf{Hit}\colon \{0,1\}^\ell \times \{0,1\}^q \to \{0,1\}^s$ such that for any set $E \subseteq \{0,1\}^s$ with $|E| \ge \beta\cdot 2^s$,*

$$\Pr_{x\in\{0,1\}^\ell}[\exists y, \mathsf{Hit}(x,y)\in E] \ge 1-\gamma.$$

Let $G\colon \{0,1\}^s \to \{0,1\}^n$ be a given $(\frac{1}{4n^2})$-PRG for width-w length-n ROBPs, and let

$\mathsf{Hit}\colon \{0,1\}^\ell \times \{0,1\}^q \to \{0,1\}^s$ be a $(\frac{1}{4n^2}, \frac{1}{2wn})$-hitter. For positive integers $t \le \frac{\log(1/\varepsilon)}{\log n} + 1$ and $n_1, \dots, n_t$ with $n_1 + \cdots + n_t = n$, our HSG G' is given by

$$G'(x, t, y_1, \dots, y_t, n_1, \dots, n_t) = G(\mathsf{Hit}(x, y_1))_{1\dots n_1} \mathbin{+\!\!+} \cdots \mathbin{+\!\!+} G(\mathsf{Hit}(x, y_t))_{1\dots n_t} \in \{0,1\}^n,$$

where $+\!\!+$ denotes string concatenation.

Correctness

Claim 3.1.5. *Let f be a width-w length-n ROBP with $\mathbb{E}[f] \ge \varepsilon$. There exist x, t, $y_1, \dots, y_t$, and $n_1, \dots, n_t$ such that $f(G'(x, t, y_1, \dots, y_t, n_1, \dots, n_t)) = 1$.*

Proof. For each vertex v, we define a "target" set $T(v)$ of vertices as follows. If $p_{v\to} \ge \frac{1}{2n^2}$, then we let $T(v) = \{v_{\text{acc}}\}$. Otherwise, we get $T(v)$ by applying Lemma 3.1.3 to the ROBP $f_{v\to}$ with $\lambda = n$. Either way, $p_{v\to T(v)} \ge \frac{1}{2n^2}$. Therefore,

$$\mathbb{E}[f_{v\to T(v)}(G(U_s))] \ge \frac{1}{4n^2}.$$

By the hitter condition, when we pick $x \in \{0,1\}^\ell$ uniformly at random, except with probability $\frac{1}{2wn}$,

$$\exists y,\ f_{v\to T(v)}(G(\mathsf{Hit}(x, y))) = 1. \tag{3.1}$$

By the union bound, there is some x such that Eq. (3.1) holds for all v simultaneously.

Now, we will inductively identify the values $y_1, \dots$ and $n_1, \dots$, along with vertices $v_{\text{start}} = v_0, v_1, \dots, v_t = v_{\text{acc}}$. Let $i \ge 1$, and assume we have already defined $y_1, \dots, y_{i-1}$, $n_1, \dots, n_{i-1}$, and $v_0, \dots, v_{i-1}$. Let n_i be the distance from v_{i-1} to $T(v_{i-1})$, let y_i be the value y of Eq. (3.1) for $v = v_{i-1}$, and let v_i be the vertex in $T(v_{i-1})$ reached by starting at v_{i-1} and reading the first n_i bits of $G(\mathsf{Hit}(x, y_i))$. When $v_i = v_{\text{acc}}$, we terminate the process and set $t = i$.

By construction, f accepts $G(\mathsf{Hit}(x,y_1))_{1\dots n_1} \# \cdots \# G(\mathsf{Hit}(x,y_t))_{1\dots n_t}$, because it passes through each v_i in sequence, eventually reaching $v_t = v_{\text{acc}}$. Now let us bound t. Whenever $v_i \neq v_{\text{acc}}$, by Lemma 3.1.3, we have $p_{v_i \to} \geq n \cdot p_{v_{i-1} \to}$. Therefore, $p_{v_{t-1} \to} \geq n^{t-1}\varepsilon$, so $n^{t-1}\varepsilon \leq 1$ and hence $t \leq \frac{\log(1/\varepsilon)}{\log n} + 1$. □

Seed length

For G, we plug in Armoni's generator (Theorem 2.5.1). For error $\frac{1}{4n^2}$, it has seed length

$$s = O\left(\frac{\log(wn)\log n}{\max\{1, \log\log w - \log\log n\}}\right).$$

For Hit, we plug in the sampling algorithm by Bellare, Goldreich, and Goldwasser [BGG93]. For threshold $\frac{1}{4n^2}$ and failure probability $\frac{1}{2wn}$, its input lengths are $\ell = O(s + \log(wn))$ and $q = O(\log n)$. Therefore, the overall seed length of G' is bounded by

$$\begin{aligned}
&\underbrace{\ell}_{\text{for } x} + \underbrace{O(\log\log(1/\varepsilon))}_{\text{for } t} + \underbrace{\left(\frac{\log(1/\varepsilon)}{\log n} + 1\right) \cdot (q + \log n)}_{\text{for } y_1,\dots,y_t,n_1,\dots,n_t} \\
&= O(s + \log(wn/\varepsilon)) \\
&= O\left(\frac{\log(wn)\log n}{\max\{1, \log\log w - \log\log n\}} + \log(1/\varepsilon)\right).
\end{aligned}$$

This completes the proof of Theorem 3.1.1.

3.2 Derandomization of Small-Success RL

In the last section, we presented an HSG for small-success **RL**. Any explicit HSG for ROBPs immediately implies a "black-box" derandomization of log-space algorithms with one-sided error, i.e., a derandomization that works by simply plugging strings into the relevant ROBP

and observing the outcomes, without taking into account the internal workings of the ROBP. In this section, as another application of our structural lemma (Lemma 3.1.2), we present an improved *non-black-box* derandomization of **RL** in the small-success regime. (Indeed, the parameters we will achieve are provably impossible for black-box algorithms.) We begin by more clearly defining the class we seek to derandomize.

Definition 3.2.1. *Let $\varepsilon = \varepsilon(N)$ be a function. We define* ε-success **RL** *to be the class of languages L such that there exists a randomized log-space algorithm A that always halts such that for every N, for every $x \in \{0,1\}^N$,*

$$\begin{aligned} x \in L &\implies \Pr[A(x) = 1] \geq \varepsilon(N) \\ x \notin L &\implies \Pr[A(x) = 1] = 0. \end{aligned}$$

Note that **RL** $=$ $(1/2)$-success **RL** $=$ $(1/\operatorname{poly}(N))$-success **RL**, and at the extreme limit we reach **NL** $=$ $(2^{-\operatorname{poly}(N)})$-success **RL**. We are most interested in the intermediate regime, $2^{-\operatorname{poly}(N)} \ll \varepsilon \ll 1/\operatorname{poly}(N)$. Prior to our work, the state-of-the-art derandomization was due to Saks and Zhou. Recall that Saks and Zhou proved that **BPSPACE**(S) is contained in **DSPACE**$(S^{3/2})$ [SZ99]. More generally, given a width-n length-n ROBP f and $\varepsilon > 0$, they showed how to deterministically estimate $\mathbb{E}[f]$ to within $\pm\varepsilon$ in space

$$O\left(\log^{3/2} n + \sqrt{\log n} \cdot \log(1/\varepsilon)\right).$$

An immediate consequence of their work is therefore

$$(\varepsilon\text{-success } \mathbf{RL}) \subseteq \mathbf{DSPACE}\left(\log^{3/2} N + \sqrt{\log N} \cdot \log(1/\varepsilon)\right).$$

We now present a better bound:

Theorem 3.2.2. *Let $\varepsilon = \varepsilon(N)$ be a function, and suppose $\lceil \log(1/\varepsilon) \rceil$ is constructible in space $O(\log^{3/2} N/\sqrt{\log\log N} + \log N \cdot \log\log(1/\varepsilon))$. Then*

$$(\varepsilon\text{-success } \mathbf{RL}) \subseteq \mathbf{DSPACE}\left(\frac{\log^{3/2} N}{\sqrt{\log\log N}} + \log N \cdot \log\log(1/\varepsilon)\right),$$

where N denotes the input length.

Our bound is better in two respects: the constant-success term (we shave off a factor of $\sqrt{\log\log N}$ from Saks and Zhou's $\log^{3/2} N$ term) and the small-success term ($\sqrt{\log N} \cdot \log(1/\varepsilon)$ is replaced by $\log N \cdot \log\log(1/\varepsilon)$). Regarding the constant-success term, for now, let us take for granted the following result, which we will prove in Section 4.2 using WPRGs.

Theorem 3.2.3 (see Theorem 4.2.11)**.** *Given a width-n length-n ROBP f, it is possible to estimate $\mathbb{E}[f]$ to within $\pm 1/n^4$ deterministically in space $O\left(\log^{3/2} n/\sqrt{\log\log n}\right)$.*

Our focus in this section is the small-success term. Our improved derandomization (Theorem 3.2.2) has a *doubly logarithmic* dependence on ε. Whereas the Saks-Zhou algorithm requires $\varepsilon \geq 1/\operatorname{poly}(N)$ to run in space $O(\log^{3/2} N)$, our algorithm runs in space $O(\log^{3/2} N)$ even when ε is as small as $2^{-2^{\Theta(\sqrt{\log N})}}$. As ε gets smaller still, our algorithm smoothly interpolates between the $O(\log^{3/2} N/\sqrt{\log\log N})$ space bound and Savitch's classic theorem $\mathbf{NL} \subseteq \mathbf{DSPACE}(\log^2 N)$ [Sav70]. The improved ε-dependence is joint work with David Zuckerman [HZ20][2].

Proof of Theorem 3.2.2. Given a width-n length-n ROBP and $\varepsilon > 0$, we will show how to distinguish $\mathbb{E}[f] \geq \varepsilon$ vs. $\mathbb{E}[f] = 0$, which will suffice by Proposition 2.4.1. Let $V = V_0 \cup \cdots \cup V_n$ be the vertices of f, and define a "shortcut graph" $G = (V, E)$ as follows. For every $i < j$ and every $u \in V_i, v \in V_j$, we include an edge (u, v) in G if the algorithm of Theorem 3.2.3

[2]In our paper [HZ20], we only achieved space complexity $O(\log^{3/2} N + \log N \cdot \log\log(1/\varepsilon))$, because the simulation of **BPL** in space $o(\log^{3/2} N)$ came later [CDRSTS21; PV21; Hoz21b].

returns a value that is greater than $1/n^4$ when we ask it to estimate $p_{u\to v}$. We use Savitch's algorithm [Sav70] to check if there is a path in G from v_{start} to v_{acc} of length at most $\lceil \log(1/\varepsilon) \rceil + 1$. If so, we accept, otherwise we reject.

Clearly, if there is an edge (u, v) in G, then v is reachable from u in f. Therefore, if $\mathbb{E}[f] = 0$, we will reject. Now suppose $\mathbb{E}[f] \geq \varepsilon$. Define $v_{\text{start}} = v_0, v_1, \ldots, v_t = v_{\text{acc}}$, where v_{i+1} is obtained from applying Lemma 3.1.2 to $f_{v_i\to}$ with $\lambda = n$ until we reach a vertex v_{t-1} with $p_{v_{t-1}\to} > 1/n$ (and then we just set $v_t = v_{\text{acc}}$). By Lemma 3.1.2, $p_{v_{i-1}\to v_i} \geq 0.5/n^3$, so the estimate for $p_{v_{i-1}\to v_i}$ that we get from Theorem 3.2.3 will be at least $0.5/n^3 - 1/n^4 > 1/n^4$ and the edge (v_{i-1}, v_i) will be included in G. Furthermore, Lemma 3.1.2 guarantees that $p_{v_{t-1}} \geq n^{t-1} \cdot \varepsilon$, so $n^{t-1} \cdot \varepsilon \leq 1$ and hence $t \leq \frac{\log(1/\varepsilon)}{\log n} + 1 < \lceil \log(1/\varepsilon) \rceil + 1$.[3] Thus, our algorithm will accept.

Our algorithm uses $O(\log n \cdot \log\log(1/\varepsilon))$ bits of space for Savitch's algorithm, plus $O(\log^{3/2} n/\sqrt{\log\log n})$ bits of space to run the algorithm of Theorem 3.2.3 each time we need to check whether an edge is present in G. □

Subsequent to our work [HZ20], Theorem 3.2.2 was generalized by Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan [AKMPSV20] to the two-sided case. That is, they designed an algorithm for computing the expectation of a width-n length-n ROBP to with $\pm\varepsilon$ (just like Theorem 3.2.3). Combining their methods with Theorem 3.2.3, their algorithm can be made to run in the same space bound we achieve for small-success **RL**, i.e., $O(\log^{3/2} n/\sqrt{\log\log n} + \log n \cdot \log\log(1/\varepsilon))$.

[3] The distinction $\log(1/\varepsilon)$ vs. $\frac{\log(1/\varepsilon)}{\log n}$ makes no difference in our final space bound because it is drowned out by the $\log^{3/2} n/\sqrt{\log\log n}$ term.

3.3 Optimal HSGs Would Derandomize BPL

In Section 3.1, we saw an HSG construction that beats the best known PRGs for ROBPs in some cases, confirming that HSGs can be easier to construct than PRGs. On the flip side, HSGs are considered less valuable than PRGs. The most basic reason for this judgment is that naïvely, HSGs only imply derandomization of algorithms with *one-sided error*, whereas PRGs can be used to derandomize algorithms with two-sided error. In the context of space-bounded computation, optimal HSGs for ROBPs would naïvely merely imply $\mathbf{L} = \mathbf{RL}$, whereas optimal PRGs for ROBPs would imply $\mathbf{L} = \mathbf{BPL}$.

In this section, by doing something less naïve than trying all seeds, we will prove that optimal HSGs for ROBPs actually *can* be used to derandomize algorithms with two-sided error. This is joint work with Kuan Cheng [CH20].

Theorem 3.3.1 (Joint with Kuan Cheng [CH20])**.** *Assume that for every $n \in \mathbb{N}$, there is an explicit $\frac{1}{2}$-HSG for width-n length-n ROBPs with seed length $O(\log n)$. Then $\mathbf{L} = \mathbf{BPL}$.*

Theorem 3.3.1 shows that HSGs are more valuable than they appear. Paired with our HSG construction in Section 3.1, this makes the HSG approach to derandomization look quite promising.

3.3.1 Context: Using HSGs to Derandomize BPP

Before proving Theorem 3.3.1 (regarding the derandomization of **BPL** using an HSG), let us discuss the well-studied analogue for time complexity. HSGs for circuits would trivially derandomize **RP**. In the 90s, it was discovered that they could also be used to derandomize **BPP**.

Theorem 3.3.2 ([ACR98])**.** *Assume that for every $n \in \mathbb{N}$, there is a $\frac{1}{2}$-HSG for size-n circuits on n input bits with seed length $O(\log n)$ that can be computed in $\mathrm{poly}(n)$ time.*

Then $\mathbf{P} = \mathbf{BPP}$.

Theorem 3.3.2 was first proven by Andreev, Clementi, and Rolim [ACR98] using a tricky iterative compression strategy. Alternative proofs were discovered later, and at this point, no fewer than *five* distinct proofs have been published [ACR98; ACRT99; BF99; GVW11; CH20]. We will present a proof due to Kuan Cheng and the author [CH20], the main virtue of which is its relative simplicity. We start with a generic lemma showing how to use hitting sets to distinguish $\mathbb{E}[f] \approx 0$ vs. $\mathbb{E}[f] \approx 1$.

Lemma 3.3.3. *Let $n \in \mathbb{N}$, let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^n \to \{0,1\}$ that is closed under complementation and shifts, let $H \subseteq \{0,1\}^n$ be a $\frac{1}{2}$-hitting set for $\mathcal{F}$, and let $H' \subseteq \{0,1\}^n$ be a $\frac{1}{2}$-hitting set for $\mathsf{AND}_{|H|} \circ \mathcal{F}$. Assume that every $f \in \mathcal{F}$ is $\left(\frac{1}{2|H|}\right)$-close to a constant. Then for every $f \in \mathcal{F}$,*

$$\mathbb{E}[f] > \frac{1}{2} \iff \exists x \in H', \forall y \in H, f(x \oplus y) = 1.$$

Proof. First, suppose $\mathbb{E}[f] \leq 1/2$. Fix $x \in H'$ and define $g(y) = \neg f(x \oplus y)$. Then $g \in \mathcal{F}$ and $\mathbb{E}[g] \geq 1/2$, so H hits g, i.e., there is some $y \in H$ such that $f(x \oplus y) = 0$.

Conversely, suppose $\mathbb{E}[f] > 1/2$. Since f is $\left(\frac{1}{2|H|}\right)$-close to a constant, we must have $\mathbb{E}[f] \geq 1 - \frac{1}{2|H|}$. Let $h(x) = \bigwedge_{y \in H} f(x \oplus y) \in \mathsf{AND}_{|H|} \circ \mathcal{F}$. For each $y \in H$, if we pick $x \in \{0,1\}^n$ uniformly at random, $\Pr_x[f(x \oplus y) = 0] \leq \frac{1}{2|H|}$. Therefore, by the union bound, $\mathbb{E}[h] \geq 1/2$, hence there is some $x \in H'$ such that $h(x) = 1$. □

Proof of Theorem 3.3.2. By straightforward amplification, it suffices to consider a randomized poly(N)-time decision algorithm A with failure probability 2^{-N}, where N is the input length. For each input $z \in \{0,1\}^N$, the function $A_z(x) \stackrel{\text{def}}{=} A(z,x)$ can be computed by a circuit of size $n = \text{poly}(N)$. Let $H \subseteq \{0,1\}^n$ be a $\frac{1}{2}$-hitting set for circuits of size[4] n, and let

[4]We are considering the "size" of a circuit to be the number of AND and OR gates.

$H' \subseteq \{0,1\}^n$ be a $\frac{1}{2}$-hitting set for circuits of size $n \cdot |H|$. We accept z if and only if there is some $x \in H'$ such that for all $y \in H$, $A(z, x \oplus y) = 1$.

The algorithm runs in time $\text{poly}(n, |H|, |H'|) = \text{poly}(N)$. Each function A_z is (2^{-N})-close to a constant, and since $|H| = \text{poly}(N)$, we have $2^{-N} < \frac{1}{2|H|}$ for all sufficiently large N. Therefore, the algorithm is correct by Lemma 3.3.3. □

Unfortunately, Lemma 3.3.3 is not sufficient for derandomizing **BPL** using an HSG for ROBPs. The difficulty is the function $h(x) = \wedge_{y \in H} f(x \oplus y)$ that appears in the proof of Lemma 3.3.3. Even if f can be computed by a bounded-width ROBP, it seems that h cannot, because computing $h(x)$ in small space seems to require reading x many times (once for each element of H.) The upshot is that we could use Lemma 3.3.3 to derandomize **BPL** assuming an optimal log-space HSG for polynomial-size *read-many* branching programs, but that is a very strong assumption. More or less the same issue comes up in each of the other known proofs that hitting sets derandomize **BPP** [ACR98; ACRT99; BF99; GVW11].

We will therefore need a new approach to prove Theorem 3.3.1. Unlike the derandomizations of **BPP** using a hitting set, we will take a *non-black-box* approach, i.e., our algorithm will inspect the transitions of a given ROBP f to compute its expectation rather than merely plugging in strings.

3.3.2 Local Consistency

Recall that for an ROBP f, we let $p_{u \to v}$ denote the probability of reaching v when starting at u and reading random bits. To estimate $\mathbb{E}[f]$, it obviously suffices to compute estimates $\widehat{p}_{u \to v}$ for the values $p_{u \to v}$ that are *accurate* as defined below.

Definition 3.3.4. *Let f be an ROBP with layers $V = V_0 \cup \cdots \cup V_n$. Let $u \in V_i$, and let $\widehat{p}_{u \to v} \in \mathbb{R}$ for all $v \in V$. We think of $\widehat{p}_{u \to v}$ as an estimate for $p_{u \to v}$. We define the* error

$\mathrm{Err}_{u\to v}$ *by*

$$\mathrm{Err}_{u\to v} = p_{u\to v} - \widehat{p}_{u\to v}.$$

We say that these estimates are α-accurate *if for every* $v \in V$*, we have* $|\,\mathrm{Err}_{u\to v}\,| \le \alpha$*. We say the estimates are* α-accurate in ℓ_1 norm *if for every* $j \in \{0,\dots,n\}$*, we have*

$$\sum_{v\in V_j} |\,\mathrm{Err}_{u\to v}\,| \le \alpha.$$

Our algorithm for estimating $\mathbb{E}[f]$ using an HSG is based on a notion of *local consistency*, which is an alternative way of measuring the quality of estimates $\widehat{p}_{u\to v}$. The definition is motivated by the observation that for every $u \in V_i$ and $v \in V_j$ with $j > i$,

$$p_{u\to v} = \sum_{t\in V_{j-1}} p_{u\to t}\cdot p_{t\to v}.$$

Definition 3.3.5. *Let* f *be an ROBP with layers* $V = V_0 \cup \cdots \cup V_n$*. Let* $u \in V_i$*, and let* $\widehat{p}_{u\to v} \in \mathbb{R}$ *for all* $v \in V$*. We think of* $\widehat{p}_{u\to v}$ *as an estimate for* $p_{u\to v}$*. For* $v \in V_j$ *with* $j > i$*, we define the* local consistency error $\mathrm{LCErr}_{u\to v}$ *by*

$$\mathrm{LCErr}_{u\to v} = \left(\sum_{t\in V_{j-1}} \widehat{p}_{u\to t}\cdot p_{t\to v}\right) - \widehat{p}_{u\to v}.$$

We extend the definition by setting $\mathrm{LCErr}_{u\to v} = 0$ *when* $j \le i$*. We say that the estimates are* locally δ-consistent *if for every* $v \in V$*, we have* $|\,\mathrm{LCErr}_{u\to v}\,| \le \delta$*. We say that estimates are* locally δ-consistent in ℓ_1 norm *if for every* $j \in \{0,\dots,n\}$*,*

$$\sum_{v\in V_j} |\,\mathrm{LCErr}_{u\to v}\,| \le \delta.$$

Local consistency is related to approximately inverting random-walk Laplacian matrices [Hoz21b, Appendix A]. It is easier to verify local consistency than it is to verify accuracy, yet the two quality measures are actually equivalent (up to some loss in the error parameter):

Lemma 3.3.6. *Let f be a length-n ROBP with layers $V = V_0 \cup \cdots \cup V_n$. Let $u \in V_i$, and let $\widehat{p}_{u\to v} \in \mathbb{R}$ be an estimated value for $p_{u\to v}$ for all $v \in V$. Assume that in each degenerate case $v \in V_j$ with $j \le i$, we have $\widehat{p}_{u\to v} = p_{u\to v}$.*

1. *For each $j \in [n]$ and $v \in V_j$, we have*

$$\mathrm{LCErr}_{u\to v} = \mathrm{Err}_{u\to v} - \sum_{t\in V_{j-1}} \mathrm{Err}_{u\to t} \cdot p_{t\to v}. \tag{3.2}$$

   Moreover, if the estimates are α-accurate in ℓ_1 norm, then they are locally (2α)-consistent in ℓ_1 norm.

2. *For each $v \in V$, we have*

$$\mathrm{Err}_{u\to v} = \sum_{t\in V} \mathrm{LCErr}_{u\to t} \cdot p_{t\to v}. \tag{3.3}$$

   Moreover, if the estimates are locally δ-consistent in ℓ_1 norm, then they are $(n\delta)$-accurate in ℓ_1 norm.

Proof. Let $j \in [n]$ and $v \in V_j$. If $j \le i$, then Eq. (3.2) holds because both sides are 0. If

$j > i$, then

$$\begin{aligned}
\mathrm{LCErr}_{u\to v} &= \left(\sum_{t\in V_{j-1}} \widehat{p}_{u\to t}\cdot p_{t\to v}\right) - \widehat{p}_{u\to v} \\
&= \left(\sum_{t\in V_{j-1}} (p_{u\to t} - \mathrm{Err}_{u\to t})\cdot p_{t\to v}\right) - (p_{u\to v} - \mathrm{Err}_{u\to v}) \\
&= \mathrm{Err}_{u\to v} - \sum_{t\in V_{j-1}} \mathrm{Err}_{u\to t}\cdot p_{t\to v}.
\end{aligned}$$

Supposing the estimates are α-accurate in ℓ_1 norm, we get

$$\begin{aligned}
\sum_{v\in V_j} |\,\mathrm{LCErr}_{u\to v}\,| &\le \sum_{v\in V_j} |\,\mathrm{Err}_{u\to v}\,| + \sum_{v\in V_j}\sum_{t\in V_{j-1}} |\,\mathrm{Err}_{u\to t}\,|\cdot p_{t\to v} \\
&\le \alpha + \sum_{t\in V_{j-1}} |\,\mathrm{Err}_{u\to t}\,|\cdot \sum_{v\in V_j} p_{t\to v} \\
&= \alpha + \sum_{t\in V_{j-1}} |\,\mathrm{Err}_{u\to t}\,| \\
&\le 2\alpha.
\end{aligned}$$

Next we will prove Eq. (3.3) by induction on $j - i$. When $j \le i$, Eq. (3.3) holds because

both sides are 0. For $j > i$, we have

$$\begin{aligned}
\mathrm{Err}_{u\to v} &= p_{u\to v} - \widehat{p}_{u\to v} \\
&= \mathrm{LCErr}_{u\to v} + \sum_{r\in V_{j-1}} (p_{u\to r} - \widehat{p}_{u\to r}) \cdot p_{r\to v} \\
&= \mathrm{LCErr}_{u\to v} + \sum_{r\in V_{j-1}} \mathrm{Err}_{u\to r} \cdot p_{r\to v} \\
&= \mathrm{LCErr}_{u\to v} + \sum_{r\in V_{j-1}} \sum_{t\in V} \mathrm{LCErr}_{u\to t} \cdot p_{t\to r} \cdot p_{r\to v} && \text{(Induction)} \\
&= \mathrm{LCErr}_{u\to v} + \sum_{t\in V} \mathrm{LCErr}_{u\to t} \cdot \sum_{r\in V_{j-1}} p_{t\to r} \cdot p_{r\to v} \\
&= \mathrm{LCErr}_{u\to v} + \sum_{t\in V\setminus\{v\}} \mathrm{LCErr}_{u\to t} \cdot p_{t\to v} \\
&= \sum_{t\in V} \mathrm{LCErr}_{u\to t} \cdot p_{t\to v}.
\end{aligned}$$

Supposing the estimates are locally δ-consistent in ℓ_1 norm, we get

$$\sum_{v\in V_j} |\,\mathrm{Err}_{u\to v}\,| \le \sum_{t\in V} |\,\mathrm{LCErr}_{u\to t}\,| \cdot \sum_{v\in V_j} p_{t\to v} = \sum_{t\in V} |\,\mathrm{LCErr}_{u\to t}\,| \le n\delta. \qquad \square$$

3.3.3 Locally Consistent Probability Estimates from an HSG

Let f be a given width-w length-n ROBP with vertices $V = V_0 \cup \cdots \cup V_n$, and let $\varepsilon > 0$. For a value $r = \widetilde{O}(w^2n^2/\varepsilon^2)$, we will show how to use a hitting set $H \subseteq \{0,1\}^{|V|\cdot nr}$ to estimate $\mathbb{E}[f] \pm \varepsilon$. We will think of each string $x \in \{0,1\}^{|V|\cdot nr}$ as consisting of a list of $|V|$ *segments*; for each vertex $v \in V$, there is a segment $x[v] \in \{0,1\}^{nr}$.

Intuitively, the purpose of $x[v]$ is to help us to estimate $p_{\to v}$. We consider $x[v]$ to be a list of r *sample inputs*, $x[v] = (x[v,1], \ldots, x[v,r])$, where $x[v,i] \in \{0,1\}^n$. We define $\widehat{p}_{\to v}(x)$

to be the fraction of $i \in [r]$ such that $f(x[v,i])$ visits v, i.e.,

$$\widehat{p}_{\to v}(x) = \frac{1}{r}\sum_{i=1}^{r} f_{\to v}(x[v,i]).$$

When x is clear from context, we will just write $\widehat{p}_{\to v}$. Our algorithm for estimating $\mathbb{E}[f] \pm \varepsilon$ using a hitting set H is: Find a string $x \in H$ such that the estimates $\widehat{p}_{\to v}(x)$ are locally (ε/n)-consistent in ℓ_1 norm, and then output $\widehat{p}_{\to v_{\mathrm{acc}}}(x)$.

Correctness

To show that our algorithm is correct, we must show that there is guaranteed to *exist* some $x \in H$ such that the estimates are locally (ε/n)-consistent in ℓ_1 norm. For such an x, it follows immediately from Lemma 3.3.6 that $|\widehat{p}_{\to v_{\mathrm{acc}}} - \mathbb{E}[f]| \le \varepsilon$. We begin by noting that a random x gives accurate estimates.

Lemma 3.3.7. *Let $\gamma = \frac{\varepsilon}{2wn}$. For a uniform random $x \in \{0,1\}^{|V|\cdot nr}$,*

$$\Pr\left[\text{the estimates } \widehat{p}_{\to v}(x) \text{ are } \gamma\text{-accurate}\right] \ge 1/2.$$

Proof. For a fixed vertex v, the value $\widehat{p}_{\to v}(x)$ is the average of r independent Bernoulli random variables, each of which is 1 with probability $p_{\to v}$. By Hoeffding's inequality,

$$\Pr_x[|\widehat{p}_{\to v}(x) - p_{\to v}| > \gamma] \le 2\exp(-2\gamma^2 r) \le \frac{1}{2\cdot(n+1)\cdot w}$$

by a suitable choice of $r = O(\gamma^{-2}\log(wn)) = \widetilde{O}(w^2n^2/\varepsilon^2)$. The union bound completes the proof. □

Furthermore, we now observe that accuracy can be checked by a polynomial-width ROBP. Note that we are not giving a log-space uniform *construction* of the ROBP (such a

construction would imply **BPL** = **ZPL**), but rather just proving that the ROBP exists. In particular, the ROBP has the values $p_{\to v}$ "hard-coded" into it.

Lemma 3.3.8. *For any $\gamma > 0$, there is an ROBP g of width $w \cdot (r+1)+1$ and length $|V| \cdot nr$ such that $g(x) = 1 \iff$ the estimates $\widehat{p}_{\to v}(x)$ are γ-accurate.*

Proof sketch. For each vertex $v \in V$ in sequence, the ROBP g simulates f on each successive sample input and counts how many simulations visit the vertex v. Since f has width w and the number is in $\{0, 1, \ldots, r\}$, this uses $w \cdot (r+1)$ states. If the number is outside the hard-coded range $(p_{\to v} \pm \gamma) \cdot r$, the ROBP moves to a special reject state $\perp$; in the final layer, all the states other than $\perp$ are merged into a single accept state. □

Corollary 3.3.9. *If H is a $\frac{1}{2}$-hitting set for ROBPs of width $w \cdot (r+1)+1$, then there is some $x \in H$ such that the estimates $\widehat{p}_{\to v}$ are locally (ε/n)-consistent in ℓ_1 norm.*

Proof. From Lemmas 3.3.7 and 3.3.8, it follows immediately that there is some $x \in H$ such that the estimates $\widehat{p}_{\to v}$ are $\left(\frac{\varepsilon}{2wn}\right)$-accurate, where $\gamma = \frac{\varepsilon}{2wn}$. This implies that the estimates are $\left(\frac{\varepsilon}{2n}\right)$-accurate in ℓ_1 norm. In turn, this implies by Lemma 3.3.6 that the estimates are locally (ε/n)-consistent in ℓ_1 norm. □

This completes the proof of correctness of our algorithm.

Efficiency

Claim 3.3.10. *Assume that for every $n \in \mathbb{N}$, there is a $\frac{1}{2}$-HSG for width-n length-n ROBPs with seed length $O(\log n)$ that can be computed in $O(\log n)$ space. Then our algorithm for estimating $\mathbb{E}[f] \pm \varepsilon$ given a width-w length-n ROBP f and $\varepsilon > 0$ runs in space $O(\log(wn/\varepsilon))$.*

Proof sketch. Each value $\widehat{p}_{\to v}(x)$ can be computed in $O(\log(wn/\varepsilon))$ space just by simulating f. When u and v are in adjacent layers, the values $p_{u \to v}$ can be computed trivially, as each

such value is simply 0, 1/2, or 1 depending on the number of edges from u to v. Therefore, the local consistency errors $\mathrm{LCErr}_{u\to v}$ can be computed in $O(\log(wn/\varepsilon))$ space, and hence local (ε/n)-consistency in ℓ_1 norm can be checked in $O(\log(wn/\varepsilon))$ space and our algorithm as a whole runs in the same space bound. □

Theorem 3.3.1 follows immediately from Claim 3.3.10 (see Proposition 2.4.1).

Chapter 4

WPRG Constructions and Applications

In this chapter,[1] we investigate the WPRG approach to derandomizing space-bounded algorithms. As a reminder, the concept of a WPRG was introduced relatively recently by Braverman, Cohen, and Garg [BCG20]. In Section 4.1, we give a construction of a WPRG for ROBPs with an optimal dependence on the error parameter ε. Then, in Section 4.2, we use WPRGs to unconditionally derandomize $\mathbf{BPSPACE}(S)$ in space $O\left(S^{3/2}/\sqrt{\log S}\right)$.

[1] For clarity, we provide here the full citation for the paper that this chapter is based on.

- [Hoz21b] William M. Hoza. *Better Pseudodistributions and Derandomization for Space-Bounded Computation.* 2021. ECCC: TR21-048

The author developed the ideas in the paper and wrote the paper.

4.1 Low-Error WPRGs for Polynomial-Width ROBPs

In this section, we present our low-error WPRG construction for ROBPs. Recall that Braverman, Cohen, and Garg constructed a WPRG for width-w length-n ROBPs with seed length

$$\widetilde{O}(\log(wn)\log n + \log(1/\varepsilon)).$$

Subsequently, Chattopadhyay and Liao [CL20] gave a simpler construction of a WPRG with the improved seed length

$$\widetilde{O}(\log(wn)\log n) + O(\log(1/\varepsilon)). \tag{4.1}$$

Then later, in two independent and simultaneous papers, Cohen, Doron, Renard, Sberlo, and Ta-Shma [CDRSTS21] and Pyne and Vadhan [PV21] gave an even simpler WPRG with the incomparable seed length

$$O(\log(wn)\log n) + \widetilde{O}(\log(1/\varepsilon)). \tag{4.2}$$

We now present our WPRG, which improves the seed length slightly further (see also Table 1.1).

Theorem 4.1.1 ([Hoz21b])**.** *For every $w, n \in \mathbb{N}$ and every $\varepsilon > 0$, there exists an explicit ε-WPRG for width-w length-n ROBPs with seed length*

$$O(\log(wn)\log n + \log(1/\varepsilon)).$$

Furthermore, the WPRG is $\mathrm{poly}(1/\varepsilon)$*-bounded.*

Compared to Eqs. (4.1) and (4.2), our seed length is the best of both worlds. Ours is the first WPRG for width-n length-n ROBPs with error $n^{-\log n}$ and seed length $O(\log^2 n)$.

Furthermore, in the width-n length-n case, our construction completes this line of WPRG research that has focused on the error dependence [BCG20; CL20; CDRSTS21; PV21; Hoz21b] in the sense that further improvements would require beating Nisan's generator [Nis92] in the more traditional constant-error setting. (After all, even an HSG must have seed length $\Omega(\log(1/\varepsilon))$.)

In the regime $n \ll w$, our WPRG seed length is slightly inferior to our HSG seed length (Theorem 3.1.1). For example, when $n = \operatorname{polylog} w$, it is still an open problem to construct a WPRG that matches our HSG's optimal seed length $O(\log(w/\varepsilon))$. Such a WPRG would be superior to the classic Nisan-Zuckerman PRG [NZ96] for relatively small ε such as $\varepsilon = 1/w$.

Like our HSG construction, our WPRG construction (Theorem 4.1.1) works by a generic reduction. We will show how to convert a $(1/\operatorname{poly}(wn))$-PRG for width-$w$ length-n ROBPs with seed length s into an ε-WPRG with seed length $s+O(\log(wn/\varepsilon))$, for any $\varepsilon > 0$. We achieve the claimed seed length by plugging in Nisan's PRG [Nis92]. More generally, for any α in the range $\frac{1}{6n^2} \le \alpha \le \frac{1}{6wn^2}$, we will show how to convert an α-PRG with seed length s into an ε-WPRG with seed length

$$s + O\left(\log(wn/\varepsilon) + \frac{\log w \cdot \log(1/\varepsilon)}{\log(1/\alpha)}\right).$$

Our reduction builds on the prior works by Cohen, Doron, Renard, Sberlo, and Ta-Shma [CDRSTS21] and Pyne and Vadhan [PV21]. They showed[2] how to convert a $(1/\operatorname{poly}(n))$-PRG with seed length s into an ε-WPRG with seed length

$$s + O(\log(w/\varepsilon) \cdot \log \log_n(1/\varepsilon)).$$

[2]The two papers [CDRSTS21; PV21] give almost the same construction and analysis, but the analysis by Cohen et al. [CDRSTS21] is slightly tighter, leading to the parameters quoted here.

Our reduction gives better results when starting with a PRG with error $1/\operatorname{poly}(wn)$, but in general our reduction is incomparable to theirs. The two reductions use similar techniques, as we will explain in more detail after establishing some notation and operations.

4.1.1 Operations on Pseudodistributions

A PRG is to a probability distribution as a WPRG is to a *pseudodistribution*. For our purposes, a pseudodistribution is a generalization of a probability distribution where "probabilities" are replaced by pseudoprobabilities, which are arbitrary real numbers that do not necessarily sum to 1.

Definition 4.1.2. *A* pseudodistribution *over $\{0,1\}^n$ is a formal real linear combination of n-bit strings, $A = \sum_{i=1}^{R} a_i \cdot x^{(i)}$, where $a_i \in \mathbb{R}$ and $x^{(i)} \in \{0,1\}^n$. A probability distribution is the special case that $a_i \in [0,1]$ and $\sum_{i=1}^{R} a_i = 1$.*

Our WPRG construction is based on some simple operations on pseudodistributions.

Definition 4.1.3. Scalar multiplication *of pseudodistributions is defined in the natural way:*

$$c \cdot \sum_{i=1}^{R} a_i \cdot x^{(i)} = \sum_{i=1}^{R} (c \cdot a_i) \cdot x^{(i)}.$$

Similarly, we can add *pseudodistributions over $\{0,1\}^n$ by simply thinking of the sum of sums as one big sum. The* tensor product *of pseudodistributions is given by*

$$\left(\sum_{i=1}^{R} a_i \cdot x^{(i)}\right) \otimes \left(\sum_{j=1}^{R'} b_j \cdot y^{(j)}\right) = \sum_{i=1}^{R} \sum_{j=1}^{R'} (a_i b_j) \cdot (x^{(i)} + y^{(j)}),$$

where $+$ denotes string concatenation. Consequently, if A is a pseudodistribution over $\{0,1\}^{n_1}$ and B is a pseudodistribution over $\{0,1\}^{n_2}$, then $A \otimes B$ is a pseudodistribution over $\{0,1\}^{n_1+n_2}$.

The natural generalization of expectation to pseudodistributions is called *pseudoexpectation*.

Definition 4.1.4. *Suppose* $A = \sum_{i=1}^{R} a_i \cdot x^{(i)}$ *is a pseudodistribution over* $\{0,1\}^n$ *and* $f\colon \{0,1\}^n \to \mathbb{R}$. *The* pseudoexpectation *of* f *under* A *is*

$$\widetilde{\mathbb{E}}[f(A)] = \sum_{i=1}^{R} a_i \cdot f(x^{(i)}).$$

We say that A fools f *with error* ε *if* $\left|\widetilde{\mathbb{E}}[f(A)] - \mathbb{E}[f]\right| \leq \varepsilon$.

Pseudoexpectation is *linear* in the pseudodistribution, i.e., if $A = A_0 + cA_1$, then

$$\widetilde{\mathbb{E}}[f(A)] = \widetilde{\mathbb{E}}[f(A_0)] + c\,\widetilde{\mathbb{E}}[f(A_1)].$$

Furthermore, under a tensor product pseudodistribution, product and pseudoexpectation can be interchanged. That is, suppose $f(x,y) = g(x) \cdot h(y)$ with $|x| = n_0$ and $|y| = n_1$, and suppose $A = A_0 \otimes A_1$, where A_i is a pseudodistribution over $\{0,1\}^{n_i}$. Then

$$\widetilde{\mathbb{E}}[f(A)] = \widetilde{\mathbb{E}}[g(A_0)] \cdot \widetilde{\mathbb{E}}[h(A_1)]. \tag{4.3}$$

This is analogous to the standard fact that product and expectation can be interchanged when dealing with independent random variables.

4.1.2 Amplifying Local Consistency

Let f be a length-n ROBP on vertex set $V = V_0 \cup \cdots \cup V_n$, and let $u \in V_i$ and $v \in V_j$ be vertices. If A is a pseudodistribution over $\{0,1\}^d$ with $d \geq j - i$, then we define

$$\widehat{p}_{u\to v}(A) = \widetilde{\mathbb{E}}[f_{u\to v}(A)],$$

i.e., $\widehat{p}_{u\to v}(A)$ is the pseudoprobability of reaching v when starting from u and reading a sample from A. (This is similar to how we used the same notation $\widehat{p}_{u\to v}(\cdot)$ in Section 3.3, but in this section the argument is a pseudodistribution rather than a long bitstring.) We define $\mathrm{Err}_{u\to v}(A) = p_{u\to v} - \widehat{p}_{u\to v}(A)$, like in Definition 3.3.4. Similarly, we define $\mathrm{LCErr}_{u\to v}(A)$ to be the local consistency error associated with the estimates $\widehat{p}_{u\to v}$, as defined in Definition 3.3.5, i.e.,

$$\mathrm{LCErr}_{u\to v}(A) = \left(\sum_{t\in V_{j-1}} \widehat{p}_{u\to t}(A)\cdot p_{t\to v}\right) - \widehat{p}_{u\to v}(A)$$

when $j > i$, and $\mathrm{LCErr}_{u\to v}(A) = 0$ when $j \le i$.

In this section, we will show that remarkably, it is possible to combine several independent samples from a pseudodistribution G over $\{0,1\}^n$ in such a way that the local consistency errors *decrease*. (Eventually we will plug in an actual probability distribution for G.) This construction is the first step of both our reduction and the prior reduction by Cohen, Doron, Renard, Sberlo, and Ta-Shma [CDRSTS21] and Pyne and Vadhan [PV21]. These prior works presented the construction in terms of preconditioned Richardson iteration and approximate inverses to a Laplacian matrix, building on work by Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan [AKMPSV20]. We will present an alternative (but equivalent) perspective in terms of local consistency that does not involve any matrices [Hoz21b].

Construction

For intuition, let us start with analogy. Let $q = \frac{1}{97}$, and suppose we wish to compute the decimal expansion of q. We can start with an estimate, $q \approx 0.01$. To measure the quality of this estimate, note that $q \cdot 97 = 1$ whereas $0.01 \cdot 97 = 0.97$. Evidently, our estimate is slightly too low. In particular, since $1 = 0.97 + 0.03$, the exact value must be $q = 0.01 + \frac{1}{97}\cdot 0.03$. To make progress, we can plug in our estimate for $\frac{1}{97}$ to conclude that

$q \approx 0.01 + 0.01 \cdot 0.03 = 0.0103$. Now we have a better estimate, i.e., a closer approximation to the true value $\frac{1}{97} = 0.010309278\ldots$

The local consistency amplification procedure is based on a similar principle. By Eq. (3.3), the exact probabilities $p_{u\to v}$ are given by

$$p_{u\to v} = \widehat{p}_{u\to v} + \sum_{t\in V} \mathrm{LCErr}_{u\to t} \cdot p_{t\to v}.$$

To get better estimates, we will effectively make the approximation

$$p_{u\to v} \approx \widehat{p}_{u\to v} + \sum_{t\in V} \mathrm{LCErr}_{u\to t} \cdot \widehat{p}_{t\to v}. \tag{4.4}$$

Now we present the actual construction. For $d \le n$, let G_d denote the pseudodistribution obtained by looking at the d-bit prefix of a sample from G. That is, if $G = \sum_{i=1}^{R} a_i \cdot x^{(i)}$, then

$$G_d = \sum_{i=1}^{R} a_i \cdot x^{(i)}_{1\ldots d}.$$

Think of U_1 as a pseudodistribution over $\{0,1\}$: $U_1 = \frac{1}{2}\cdot 0 + \frac{1}{2}\cdot 1$.[3] For $d \in [n]$, define a "local consistency error pseudodistribution" Δ_d by the rule

$$\Delta_d = G_{d-1} \otimes U_1 - G_d.$$

More generally, for $m, d \in [n]$ with $m \le d$, define

$$\Delta_d^{(m)} = \sum_{d_1+\cdots+d_m=d} \Delta_{d_1} \otimes \cdots \otimes \Delta_{d_m},$$

[3]To be clear, in the expression "$\frac{1}{2}\cdot 0 + \frac{1}{2}\cdot 1$", 0 and the last instance of 1 are mere symbols rather than real numbers. Taking a cue from quantum computing, we could have written $U_1 = \frac{1}{2}|0\rangle + \frac{1}{2}|1\rangle$.

where the sum is over all m-tuples of positive integers that sum to d. For $m \geq 1$, define a "correction term" $T^{(m)}$ by

$$T^{(m)} = \sum_{d=m}^{n} \Delta_d^{(m)} \otimes G_{n-d}.$$

For $m \geq 0$, the amplified pseudodistribution is

$$G^{(m)} = G + \sum_{i=1}^{m} T^{(i)} = G + \sum_{i=1}^{m} \sum_{d=i}^{n} \Delta_d^{(i)} \otimes G_{n-d}.$$

(Eq. (4.4) corresponds to the case $m = 1$.)

Analysis

Let us analyze the local consistency of $G^{(m)}$. In fact, we will give an *exact formula* for the local consistency errors of $G^{(m)}$ in terms of the local consistency errors of G. For a pseudodistribution A, vertices u, v, and $m \geq 1$, define

$$\mathrm{LCErr}_{u \to v}^{(m)}(A) = \sum_{u=u_0,\ldots,u_m=v} \prod_{i=1}^{m} \mathrm{LCErr}_{u_{i-1} \to u_i}(A),$$

where the sum is over all sequences of $m+1$ vertices starting at u and ending at v. Note that since we are considering *products* of m errors, $\mathrm{LCErr}_{u \to v}^{(m)}$ will decrease as m gets bigger, provided the local consistency errors of A are sufficiently small to start with. Our goal is to prove the following.

Lemma 4.1.5. *For any $u, v \in V$ and $m \geq 0$, we have* $\mathrm{LCErr}_{u \to v}(G^{(m)}) = \mathrm{LCErr}_{u \to v}^{(m+1)}(G)$.

We begin by analyzing pseudoprobabilities under $\Delta_d^{(m)}$. The definition of Δ_d suggests that these pseudoprobabilities should be related to the local consistency errors of G. For convenience, define $V_i = \varnothing$ when $i > n$.

Lemma 4.1.6. *Let $m, d \in [n]$ with $m \leq d$, let A be a pseudodistribution over $\{0,1\}^{n-d}$, let $u \in V_i$, and let $v \in V_j$. Then*

$$\widehat{p}_{u\to v}(\Delta_d^{(m)} \otimes A) = \sum_{t \in V_{i+d}} \mathrm{LCErr}_{u\to t}^{(m)}(G) \cdot \widehat{p}_{t\to v}(A). \tag{4.5}$$

Proof. We proceed by induction on m. First, we consider $m = 1$. If $j \geq i + d$, then the function $f_{u\to v}$ can be decomposed as

$$f_{u\to v}(x, y) = \sum_{t \in V_{i+d}} f_{u\to t}(x) \cdot f_{t\to v}(y),$$

where $|x| = d$. Therefore, for any pseudodistribution B over $\{0,1\}^d$, we have

$$\widehat{p}_{u\to v}(B \otimes A) = \sum_{t \in V_{i+d}} \widehat{p}_{u\to t}(B) \cdot \widehat{p}_{t\to v}(A)$$

(see Eq. (4.3)). For $t \in V_{i+d}$, clearly $\widehat{p}_{u\to t}(\Delta_d) = \mathrm{LCErr}_{u\to t}(G)$, completing the proof in this case. Meanwhile, if $i \leq j < i + d$, then $G_{d-1} \otimes U_1 \otimes A$ and $G_d \otimes A$ induce the same pseudodistribution on the first $j - i$ bits, namely G_{j-i}. Therefore, $\widehat{p}_{u\to v}(\Delta_d \otimes A) = 0$. Furthermore, for $t \in V_{i+d}$, trivially $\widehat{p}_{t\to v}(A) = 0$, so in this case Eq. (4.5) becomes $0 = 0$. Finally, if $j < i$, then Eq. (4.5) trivially becomes $0 = 0$.

Now, for the inductive step, if $1 \leq m < d$, then

$$\Delta_d^{(m+1)} = \sum_{k=m}^{d-1} \Delta_{d-k} \otimes \Delta_k^{(m)},$$

and therefore

$$\begin{aligned}
&\widehat{p}_{u\to v}(\Delta_d^{(m+1)} \otimes A) \\
&\quad = \sum_{k=m}^{d-1} \widehat{p}_{u\to v}(\Delta_{d-k} \otimes \Delta_k^{(m)} \otimes A) \\
&\quad = \sum_{k=m}^{d-1} \sum_{r\in V_{i+d-k}} \mathrm{LCErr}_{u\to r}(G) \cdot \widehat{p}_{r\to v}(\Delta_k^{(m)} \otimes A) && \text{(Base case)} \\
&\quad = \sum_{t\in V_{i+d}} \sum_{k=m}^{d-1} \sum_{r\in V_{i+d-k}} \mathrm{LCErr}_{u\to r}(G) \cdot \mathrm{LCErr}_{r\to t}^{(m)}(G) \cdot \widehat{p}_{t\to v}(A) && \text{(Induction)} \\
&\quad = \sum_{t\in V_{i+d}} \mathrm{LCErr}_{u\to t}^{(m+1)}(G) \cdot \widehat{p}_{t\to v}(A). && \square
\end{aligned}$$

Now we can prove Lemma 4.1.5, which states that $\mathrm{LCErr}_{u\to v}(G^{(m)}) = \mathrm{LCErr}_{u\to v}^{(m+1)}(G)$.

Proof of Lemma 4.1.5. By Lemma 4.1.6, if $u \in V_i$ and $v \in V_j$ and $m \in [n]$, then

$$\begin{aligned}
\widehat{p}_{u\to v}(T^{(m)}) &= \sum_{d=m}^{n} \widehat{p}_{u\to v}(\Delta_d^{(m)} \otimes G_{n-d}) \\
&= \sum_{d=m}^{n} \sum_{r\in V_{i+d}} \mathrm{LCErr}_{u\to r}^{(m)}(G) \cdot \widehat{p}_{r\to v}(G) \\
&= \sum_{r\in V} \mathrm{LCErr}_{u\to r}^{(m)}(G) \cdot \widehat{p}_{r\to v}(G).
\end{aligned}$$

Now, if $j \le i$, then the lemma is trivial, because $\mathrm{LCErr}_{u\to v}(G^{(m)}) = \mathrm{LCErr}_{u\to v}^{(m+1)}(G) = 0$.

Assume therefore that $j > i$. Then

$$\begin{aligned}\mathrm{LCErr}_{u\to v}(T^{(m)}) &= \left(\sum_{t\in V_{j-1}} \widehat{p}_{u\to t}(T^{(m)})\cdot p_{t\to v}\right) - \widehat{p}_{u\to v}(T^{(m)}) \\ &= \sum_{r\in V} \mathrm{LCErr}^{(m)}_{u\to r}(G)\cdot \underbrace{\left(\left(\sum_{t\in V_{j-1}} \widehat{p}_{r\to t}(G)\cdot p_{t\to v}\right) - \widehat{p}_{r\to v}\right)}_{(*)}.\end{aligned}$$

Expression $(*)$ should look familiar. It is the formula for $\mathrm{LCErr}_{r\to v}(G)$ in the case that r is in a layer strictly before V_j. More generally, $\mathrm{LCErr}_{r\to v}(G) = (*)$, except when $r = v$, in which case $\mathrm{LCErr}_{r\to v}(G) = 0$ while $(*) = -1$. Therefore,

$$\begin{aligned}\mathrm{LCErr}_{u\to v}(T^{(m)}) &= \left(\sum_{r\in V} \mathrm{LCErr}^{(m)}_{u\to r}(G)\cdot \mathrm{LCErr}_{r\to v}(G)\right) - \mathrm{LCErr}^{(m)}_{u\to v}(G) \\ &= \mathrm{LCErr}^{(m+1)}_{u\to v}(G) - \mathrm{LCErr}^{(m)}_{u\to v}(G).\end{aligned}$$

(The same holds for $m > n$, because in this case $\mathrm{LCErr}_{u\to v}(T^{(m)})$, $\mathrm{LCErr}^{m+1}_{u\to v}(G)$, and $\mathrm{LCErr}^{(m)}_{u\to v}(G)$ are all zero.) Overall, what this means is that we get a telescoping sum:

$$\begin{aligned}\mathrm{LCErr}_{u\to v}(G^{(m)}) &= \mathrm{LCErr}_{u\to v}(G) + \sum_{i=1}^{m} \mathrm{LCErr}_{u\to v}(T^{(i)}) \\ &= \mathrm{LCErr}^{(1)}_{u\to v}(G) + \sum_{i=1}^{m} \left(\mathrm{LCErr}^{(i+1)}_{u\to v}(G) - \mathrm{LCErr}^{(i)}_{u\to v}(G)\right) \\ &= \mathrm{LCErr}^{(m+1)}_{u\to v}(G).\end{aligned}$$

□

As a corollary, if G fools every subprogram of f with moderate error, then $G^{(m)}$ fools f with low error:

Corollary 4.1.7. *Assume that the estimates generated by G are α-accurate in ℓ_1 norm, i.e.,*

for every vertex u and every layer V_j, we have

$$\sum_{v \in V_j} |\widehat{p}_{u \to v}(G) - p_{u \to v}| \leq \alpha.$$

Then for every $m \geq 0$, $G^{(m)}$ fools f with error $n \cdot (2\alpha n)^{m+1}$.

To be clear, Corollary 4.1.7 readily follows from the analysis in the prior works of Cohen, Doron, Renard, Sberlo, and Ta-Shma [CDRSTS21] and Pyne and Vadhan [PV21]. One of our key contributions is the observation that rather than fooling *all* ROBPs, it suffices for G to fool the subprograms of f.

Proof of Corollary 4.1.7. We will show by induction on $m \geq 1$ that for any vertex $u \in V_i$ and any layer V_j, we have

$$\sum_{v \in V_j} |\operatorname{LCErr}^{(m)}_{u \to v}(G)| \leq (2\alpha n)^m. \tag{4.6}$$

For the base case, consider $m = 1$. By Lemma 3.3.6, the estimates generated by G are locally (2α)-consistent in ℓ_1 norm, so indeed, we get the bound 2α, which is at most $2\alpha n$. Now for

the inductive step, for $m \geq 1$, we have

$$\begin{aligned}
\sum_{v \in V_j} |\operatorname{LCErr}^{(m+1)}_{u \to v}(G)| &= \sum_{v \in V_j} \left| \sum_{t \in V} \operatorname{LCErr}^{(m)}_{u \to t}(G) \cdot \operatorname{LCErr}_{t \to v}(G) \right| \\
&\leq \sum_{t \in V} |\operatorname{LCErr}^{(m)}_{u \to t}(G)| \cdot \sum_{v \in V_j} |\operatorname{LCErr}_{t \to v}(G)| \\
&\leq 2\alpha \sum_{t \in V} |\operatorname{LCErr}^{(m)}_{u \to t}(G)| && \text{(Local consistency)} \\
&= 2\alpha \sum_{k=m}^{n} \sum_{t \in V_k} |\operatorname{LCErr}^{(m)}_{u \to t}(G)| \\
&\leq 2\alpha \sum_{k=m}^{n} (2\alpha n)^m && \text{(Induction)} \\
&= (2\alpha n)^{m+1},
\end{aligned}$$

completing the proof of Eq. (4.6). By Lemma 4.1.5, it follows that the estimates generated by $G^{(m)}$ are locally $(2\alpha n)^{m+1}$-consistent in ℓ_1 norm, so we are done by Lemma 3.3.6. □

4.1.3 The Reduction: Converting $(\frac{1}{\operatorname{poly}(wn)})$-PRGs into ε-WPRGs

In the last section, we saw how to amplify the local consistency of a pseudodistribution by combining several independent samples. If we applied that construction to Nisan's PRG [Nis92] directly, we would get a low-error WPRG with a large seed length because each term $\Delta_d^{(m)} \otimes G_{n-d}$ involves several independent samples from G. To reduce the seed length, we would like to use correlated seeds to Nisan's PRG.

In the prior works [CDRSTS21; PV21], the approach was to use the INW generator [INW94] to sample a sequence of pseudorandom seeds to Nisan's PRG. Because the INW generator is non-optimal, this leads to an overall seed length of $O(\log^2 n + \log(1/\varepsilon) \cdot \log \log_n(1/\varepsilon))$ for fooling width-n length-n ROBPs.

As mentioned previously, our key observation is that we do not need G to fool *all* ROBPs. The local consistency amplification procedure works as long as G fools all subprograms of f with moderate error, where f is the specific program we are trying to fool. To exploit this observation, we use an *averaging sampler*. Samplers are a two-sided version of hitters (Definition 3.1.4) and are essentially equivalent to seeded extractors [Zuc97].

Definition 4.1.8. *A function* $\mathsf{Samp}\colon \{0,1\}^\ell \times \{0,1\}^q \to \{0,1\}^s$ *is a* (β,γ)-sampler *if for every* $f\colon \{0,1\}^s \to [0,1]$*, we have*

$$\Pr_{x\sim U_\ell}\left[\left|\mathbb{E}[f] - 2^{-q}\sum_{y\in\{0,1\}^q} f(\mathsf{Samp}(x,y))\right| \le \beta\right] \ge 1-\gamma.$$

The idea is, let G denote Nisan's generator. If we pick x at random, then with high probability, $G(\mathsf{Samp}(x,\cdot))$ is a good PRG for all subprograms of f with optimal seed length $O(\log(wn))$. We can afford to apply the amplification procedure to that PRG because sampling several independent seeds of it is cheap.

In more detail, let $G\colon \{0,1\}^s \to \{0,1\}^n$ be a given PRG for width-w length-n ROBPs with some moderate error $\alpha \in [\frac{1}{6n^2w}, \frac{1}{6n^2}]$. (Eventually we will choose $\alpha = \frac{1}{6n^2w}$ to prove Theorem 4.1.1.) Let $\varepsilon > 0$ be the desired (low) error of our WPRG, and define

$$\begin{aligned} m &= \left\lceil \frac{\log(n/\varepsilon)}{\log(1/(6n\alpha))} \right\rceil = O\left(1 + \frac{\log(1/\varepsilon)}{\log n}\right), \\ \beta &= \alpha/w = \left(\frac{1}{wn}\right)^{O(1)}, \\ \gamma &= \frac{\varepsilon}{2w^2n^2\cdot((2n)^{m+2}+1)} = \left(\frac{\varepsilon}{wn}\right)^{O(1)}. \end{aligned}$$

Let $\mathsf{Samp}\colon \{0,1\}^\ell \times \{0,1\}^q \to \{0,1\}^s$ be a (β,γ)-sampler. For each $x \in \{0,1\}^\ell$, let G_x be the probability distribution $G(\mathsf{Samp}(x,U_q))$. Our low-error pseudorandom pseudodistribution

G' for width-w length-n ROBPs is obtained by sampling x uniformly at random and then amplifying the local consistency of G_x, i.e.,

$$G' = 2^{-\ell} \cdot \sum_{x \in \{0,1\}^\ell} G_x^{(m)}.$$

Correctness

Claim 4.1.9. *G' fools width-w length-n ROBPs with error ε.*

Proof. For any $x \in \{0,1\}^\ell$, the amplified pseudodistribution $G_x^{(m)}$ has the form $\sum_{i=1}^K \pm A_i$, where each A_i is a tensor product of probability distributions and

$$K \leq (m+1) \cdot n \cdot n^m \cdot 2^m \leq (2n)^{m+2}.$$

Therefore, $\left|\widetilde{\mathbb{E}}[f(G_x^{(m)})]\right| \leq (2n)^{m+2}$.

For any pair of vertices u, v, by the sampler condition, with probability $1-\gamma$ over a uniform random x, we have $|\widehat{p}_{u\to v}(G_x) - \widehat{p}_{u\to v}(G)| \leq \beta$ (identifying G with the probability distribution $G(U_s)$). Let BAD be the set of $x \in \{0,1\}^\ell$ such that there exist u, v with $|\widehat{p}_{u\to v}(G_x) - \widehat{p}_{u\to v}(G)| > \beta$. By the union bound,

$$|\mathsf{BAD}| \leq w^2 n^2 \cdot \gamma \cdot 2^\ell.$$

For $x \notin \mathsf{BAD}$, for any vertex u and any layer V_j, we have

$$\sum_{v \in V_j} |\widehat{p}_{u\to v}(G_x) - p_{u\to v}| \leq w\beta + \sum_{v \in V_j} |\widehat{p}_{u\to v}(G) - p_{u\to v}|.$$

Now, G fools width-w length-n ROBPs with error α. Therefore, for any set $T \subseteq V_j$, we have $|\widehat{p}_{u\to T}(G) - p_{u\to T}| \leq \alpha$, since we could merge the vertices in T into a single accept vertex.

Since total variation distance is half ℓ_1 distance, this implies that

$$\sum_{v \in V_j} |\widehat{p}_{u \to v}(G) - p_{u \to v}| \leq 2\alpha.$$

Therefore, for $x \notin \mathsf{BAD}$, we have

$$\sum_{v \in V_j} |\widehat{p}_{u \to v}(G_x) - p_{u \to v}| \leq w\beta + 2\alpha = 3\alpha,$$

and hence by Corollary 4.1.7, G_x fools f with error $n \cdot (6\alpha n)^{m+1} \leq \varepsilon/2$. Overall, we get the bound

$$\begin{aligned}
\left|\widetilde{\mathbb{E}}[f(G')] - \mathbb{E}[f]\right| &\leq 2^{-\ell} \cdot \left(\sum_{x \in \mathsf{BAD}} |\,\widetilde{\mathbb{E}}[f(G_x^{(m)})] - \mathbb{E}[f]| + \sum_{x \notin \mathsf{BAD}} |\,\widetilde{\mathbb{E}}[f(G_x^{(m)})] - \mathbb{E}[f]| \right) \\
&\leq 2^{-\ell} \cdot |\mathsf{BAD}| \cdot ((2n)^{m+2} + 1) + \frac{\varepsilon}{2} \\
&\leq \gamma \cdot w^2 n^2 \cdot ((2n)^{m+2} + 1) + \frac{\varepsilon}{2} \\
&= \varepsilon.
\end{aligned}$$

□

Seed length and efficiency

We presented our low-error pseudodistribution G'. To bound our seed length, we need to look at a corresponding WPRG.

Definition 4.1.10. *If (G, ρ) is a WPRG with seed length s, the pseudodistribution* sampled by (G, ρ) *is given by*

$$\sum_{x \in \{0,1\}^s} 2^{-s} \cdot \rho(x) \cdot G(x).$$

Observe that (G, ρ) fools f with error ε if and only if the pseudodistribution sampled by (G, ρ) fools f with error ε.

For each of the operations on pseudodistributions that we considered, there is a corresponding operation on WPRGs.

Definition 4.1.11. *If (G, ρ) is a WPRG and $c \in \mathbb{R}$, we define $c \cdot (G, \rho)$ to be the WPRG (G, ρ'), where $\rho'(x) = c \cdot \rho(x)$. Observe that if (G, ρ) samples the pseudodistribution A, then $c \cdot (G, \rho)$ samples $c \cdot A$. If (G, ρ) is K-bounded, then $c \cdot (G, \rho)$ is (cK)-bounded.*

Suppose that for each $b \in \{0, 1\}$, we have a WPRG (G_b, ρ_b) with seed length s, where G_b has output length n. We define the sum $(G_0, \rho_0) + (G_1, \rho_1)$ to be a WPRG (G, ρ) with seed length $s + 1$ given by

$$G(x, b) = G_b(x)$$
$$\rho(x, b) = 2 \cdot \rho_b(x).$$

(The factor of 2 accounts for the fact that in the definition of a WPRG, we consider an expectation over seeds rather than a sum.) If (G_b, ρ_b) samples the pseudodistribution A_b, then (G, ρ) samples $A_0 + A_1$. If each (G_b, ρ_b) is K-bounded, then (G, ρ) is $(2K)$-bounded.

Finally, suppose that for each $b \in \{0, 1\}$, we have a WPRG (G_b, ρ_b) with seed length s_b, where G_b has output length n_b. We define the tensor product $(G_0, \rho_0) \otimes (G_1, \rho_1)$ to be a WPRG (G, ρ) with seed length $s_0 + s_1$ given by

$$G(x, y) = G_0(x) +\!\!+ G_1(y)$$
$$\rho(x, y) = \rho_0(x) \cdot \rho_1(y),$$

where $+\!\!+$ denotes string concatenation. If (G_b, ρ_b) samples the pseudodistribution A_b, then (G, ρ) samples $A_0 \otimes A_1$. If (G_b, ρ_b) is K_b-bounded, then (G, ρ) is $(K_0 K_1)$-bounded.

We will use the following explicit sampler.

Theorem 4.1.12 ([CL20, Appendix B]). *For every $s \in \mathbb{N}$ and every $\beta, \gamma > 0$, there exists a (β, γ)-sampler $\mathsf{Samp}\colon \{0,1\}^\ell \times \{0,1\}^q \to \{0,1\}^s$ with $\ell = s + O(\log(1/\beta) + \log(1/\gamma))$ and $q = O(\log(1/\beta) + \log\log(1/\gamma))$, such that given s, β, γ, x, and y, the value $\mathsf{Samp}(x, y)$ can be computed in space $O(s + \log(1/\beta) + \log(1/\gamma))$.*

In our case, we get

$$\begin{aligned}\ell &= s + O(\log(wn/\varepsilon))\\ q &= O(\log(wn) + \log\log(1/\varepsilon)).\end{aligned}$$

Proof of Theorem 4.1.1. Elaborating on the calculation in the proof of Claim 4.1.9, $G_x^{(m)}$ is of the form $\sum_{i=1}^{K} \pm A_i$, where each A_i is a tensor product of $O(m)$ probability distributions, each of which is either a prefix of G_x or else is U_1, and $K \le (2n)^{m+2}$. Therefore, using the operations of Definition 4.1.11, the seed length of G' is

$$\begin{aligned}\ell + O(mq + \log K) &= s + O\left(\log(wn/\varepsilon) + \frac{\log(n/\varepsilon) \cdot (\log(wn) + \log\log(1/\varepsilon))}{\log(1/\alpha)}\right)\\ &= s + O\left(\log(wn/\varepsilon) + \frac{\log(n/\varepsilon) \cdot \log(wn)}{\log(1/\alpha)}\right)\\ &= s + O\left(\log(wn/\varepsilon) + \frac{\log w \cdot \log(1/\varepsilon)}{\log(1/\alpha)}\right),\end{aligned}$$

where the second step holds without loss of generality, because it holds under the assumption that $\varepsilon > 2^{-n}$, and if $\varepsilon \le 2^{-n}$, we can achieve the same seed length by simply sampling from U_n. Using the bounds in Definition 4.1.11, the WPRG is $(2K)$-bounded, where the factor of 2 is to pad the number of terms to a power of two if necessary. We may assume without loss of generality that $\varepsilon < 1/n$ (otherwise $(G, 1)$ is a suitable WPRG), so $K \le \mathrm{poly}(1/\varepsilon)$ as claimed.

Choose $\alpha = \frac{1}{6n^2 w}$, so the seed length becomes $s + O(\log(wn/\varepsilon))$. Furthermore, choose

G to be Nisan's generator, which has seed length $O(\log(wn)\log n)$ for this error α, so the overall seed length is $O(\log(wn)\log n + \log(1/\varepsilon))$. Explicitness is clear. □

4.2 Derandomization that Beats the Saks-Zhou Bound

In this section, we study the general problem of derandomizing bounded-error space-S decision algorithms that always halt, using as little space as possible. Early work [Sav70; Jun81; BCP83] implies that such algorithms (and more) can be simulated deterministically in space $O(S^2)$. Saks and Zhou improved the bound to $O(S^{3/2})$ [SZ99], a celebrated theorem that has remained unbeaten for decades. As an application of WPRGs, we will now show the slightly improved bound $O\left(S^{3/2}/\sqrt{\log S}\right)$.

Theorem 4.2.1 ([Hoz21b])**.** *For any function $S = S(N) \geq \log N$, we have*

$$\mathbf{BPSPACE}(S) \subseteq \mathbf{DSPACE}\left(S^{3/2}/\sqrt{\log S}\right).$$

We first give a high-level overview of the proof. Saks and Zhou's original algorithm [SZ99] heavily relies on Nisan's ε-PRG [Nis92] for width-w length-n ROBPs (Theorem 1.2.7), which has seed length $O(\log(wn/\varepsilon)\log n)$. Saks and Zhou set $\varepsilon = 1/\operatorname{poly}(w)$ and $n = 2^{\sqrt{\log w}} \ll w$, so the PRG has seed length $O(\log^{3/2} w)$. Saks and Zhou's main technical contribution is a method of *reusing* the seed of this PRG many times, which allows them to derandomize $(\log w)$-space algorithms even though the PRG only outputs n pseudorandom bits and the algorithm might use up to w random bits.

In the regime $n \ll w$ and $\varepsilon = 1/\operatorname{poly}(n)$, as we have discussed, Armoni [Arm98] designed a PRG that outperforms Nisan's PRG by a factor of $\Theta(\log\log w)$ (Theorem 2.5.1). Furthermore, Armoni showed how to apply Saks and Zhou's methods to any generic PRG for ROBPs in place of Nisan's PRG [Arm98]. Taken together, these results *almost* imply

a derandomization of **BPSPACE**(S) in space $o(S^{3/2})$. The catch is the error parameter. To plug a PRG into the Saks-Zhou framework, it seems necessary that the PRG have error $1/\operatorname{poly}(w)$, and for such a small error, Armoni's PRG is no better than Nisan's PRG. (On the bright side, Armoni was at least able to obtain an improved derandomization of *low-randomness* space-bounded algorithms [Arm98].)

The first step in the proof of Theorem 4.2.1 is to obtain a WPRG with error $1/\operatorname{poly}(w)$ for width-w length-n ROBPs with seed length $O(\log^{3/2} w/\sqrt{\log\log w})$, where

$$n = 2^{O(\sqrt{\log w \cdot \log\log w})}.$$

Note that such a WPRG has a slightly shorter seed length and a slightly longer output length compared to the instantiation of Nisan's PRG that Saks and Zhou used [SZ99]. Such a WPRG could be obtained by applying the reduction of Cohen, Doron, Renard, Sberlo, and Ta-Shma [CDRSTS21] and Pyne and Vadhan [PV21] to Armoni's PRG (Theorem 2.5.1), using the slightly tighter analysis by Cohen et al. [CDRSTS21]. For the sake of a unified and relatively self-contained presentation, in Section 4.2.1 we will instead obtain such a WPRG using the alternative reduction that we presented in Section 4.1.

The second step in the proof of Theorem 4.2.1 is to generalize Saks and Zhou's methods so that we can plug in a WPRG instead of an unweighted PRG. This was done already by Chattopadhyay and Liao [CL20]; we will repeat the argument in Section 4.2.2 for clarity. As this proof overview hopefully makes clear, Theorem 4.2.1 follows fairly straightforwardly from recent prior work on WPRGs and older work on derandomizing space-bounded computation. Our job is essentially just to put the pieces together.

4.2.1 Low-Error WPRGs for Short, Wide ROBPs

It is convenient to generalize ROBPs to the large-alphabet case. A *width-w length-n ROBP over the alphabet Σ* is defined just like our original ROBP definition (Definition 1.2.2), except that each vertex not in the last layer has $|\Sigma|$ outgoing edges instead of 2, with each edge labeled with a distinct symbol from Σ. The program computes a function $f\colon \Sigma^n \to \{0,1\}$ in the natural way.

Lemma 4.2.2. *For every $w \in \mathbb{N}$, there exists a K-bounded ε-WPRG (G, ρ) for width-(w^2+1) length-n ROBPs over the alphabet $\{0,1\}^a$ with seed length s, computable in space $O(s)$, where*

$$\begin{aligned} K &= \mathrm{poly}(w) & \varepsilon &= w^{-12}/24 \\ n &= \left\lceil \exp\left(\sqrt{\log w \cdot \log\log w}\right)\right\rceil & a &= \lceil 2 + 7\log w \rceil \\ s &= O\left(\log^{3/2} w / \sqrt{\log\log w}\right). \end{aligned}$$

Furthermore, ρ is integer-valued.

As mentioned previously, Lemma 4.2.2 can be proven by applying the reduction of Cohen et al. and Pyne and Vadhan [CDRSTS21; PV21] to Armoni's PRG (Theorem 2.5.1). We will give an alternative proof using our version of the error reduction procedure.

Proof of Lemma 4.2.2. It suffices to fool ROBPs of width $w' = w \cdot 2^a = \mathrm{poly}(w)$ and length $n' = n \cdot a = \widetilde{O}(n)$ over the binary alphabet, since such ROBPs can simulate width-w length-n ROBPs over the alphabet $\{0,1\}^a$. To fool such ROBPs, we use the reduction that we designed to prove Theorem 4.1.1. However, we use $\alpha = \frac{1}{6(n')^2}$ rather than $\frac{1}{6w'(n')^2}$, and we set G to be Armoni's PRG (Theorem 2.5.1) instead of Nisan's PRG. The seed length s_0 of G is

given by

$$s_0 = O\left(\frac{\log(w'n'/\alpha)\log n'}{\max\{1, \log\log(w') - \log\log(n'/\alpha)\}}\right) = O\left(\frac{\log^{3/2} w}{\sqrt{\log\log w}}\right),$$

so the seed length of the WPRG is given by

$$\begin{aligned} s_0 + O\left(\log(w'n'/\varepsilon) + \frac{\log w \cdot \log(1/\varepsilon)}{\log(1/\alpha)}\right) &= O\left(\frac{\log^{3/2} w}{\sqrt{\log\log w}} + \log w + \frac{\log^2 w}{\sqrt{\log w \cdot \log\log w}}\right) \\ &= O\left(\frac{\log^{3/2} w}{\sqrt{\log\log w}}\right). \end{aligned}$$

Finally, looking at Definition 4.1.11, it is clear that the weight function ρ is integer-valued.

□

4.2.2 Plugging WPRGs into the Saks-Zhou Framework

In this section, we show how to apply Saks and Zhou's methods [SZ99] using the WPRG of Lemma 4.2.2. This follows already from Chattopadhyay and Liao's prior work [CL20], and we include the proofs just for the sake of clarity. Most of our effort will go toward proving the following lemma.

Lemma 4.2.3. *There is a randomized algorithm that, given a width-w length-w ROBP f, outputs a value E such that*

$$\Pr[|E - \mathbb{E}[f]| \le 1/w^4] \ge 2/3.$$

The algorithm runs in $O\left(\log^{3/2} w/\sqrt{\log\log w}\right)$ space and uses $O\left(\log^{3/2} w/\sqrt{\log\log w}\right)$ random bits.

Toward proving it, fix some $w \in \mathbb{N}$. Let (G, ρ) be the K-bounded ε-WPRG for width-

(w^2+1) length-n ROBPs over the alphabet $\{0,1\}^a$ with seed length s of Lemma 4.2.2. (We will not plug in the values of K, n, or s until the very end of the proof, so the reader can see what would happen with a hypothetical better WPRG.)

Approximate matrix powering using a WPRG

We start by translating the WPRG into a method of approximating powers of substochastic matrices. This translation is very similar to what Armoni did for the case of unweighted PRGs [Arm98], which in turn is a natural generalization of what Saks and Zhou did with Nisan's PRG specifically [SZ99]. Let $\mathsf{Samp}\colon \{0,1\}^\ell \times \{0,1\}^q \to \{0,1\}^s$ be an $(\varepsilon/K, 0.1\cdot w^{-5})$-sampler.

Let $M \in [0,1]^{w^2 \times w^2}$ be a substochastic matrix where each entry has a bits of precision (i.e., each entry is an integer multiple of 2^{-a}). For random bits $x \in \{0,1\}^\ell$, we define a matrix $\mathsf{Pow}(M,x)$ that approximates M^n as follows.

1. Let $g^{(M)}$ be a width-(w^2+1) length-n ROBP over the alphabet $\{0,1\}^a$ on the layers $V_0,\dots,V_n$, where $V_i = [w^2+1]\times\{i\}$. We put $M_{u,v}\cdot 2^a$ edges going from $(u,i-1)$ to (v,i), and all the remaining edges coming from V_{i-1} lead to the special "fail state" (w^2+1,i). By construction, this ROBP satisfies $(M^n)_{u,v} = \mathbb{E}\left[g^{(M)}_{(u,0)\to(v,n)}\right]$.

2. Define

$$\mathsf{Pow}(M,x)_{u,v} = 2^{-q}\cdot \sum_{y\in\{0,1\}^q} \rho(\mathsf{Samp}(x,y))\cdot g^{(M)}_{(u,0)\to(v,n)}(G(\mathsf{Samp}(x,y))).$$

We extend the definition to the case that M is not a substochastic matrix by simply setting $\mathsf{Pow}(M,x)=0$ in this case.

Breaking correlations using randomized rounding

The rest of the construction matches Saks and Zhou's original algorithm [SZ99]. To approximate M^w, we will apply Pow many times. Crucially, we will reuse the same string x of random bits for each application of Pow. Reusing random bits here has the potential to cause serious trouble, because it means that in the second application, we will be applying Pow to a matrix that is correlated to the random bits we are using. Saks and Zhou introduced a randomized rounding scheme to break these correlations [SZ99]. In particular, for a number $\alpha \in \mathbb{R}$ and a random value $z \in [2^a]$, define

$$\mathsf{Snap}(\alpha, z) = \lfloor \alpha \cdot 2^a - z \cdot 2^{-a} \rfloor \cdot 2^{-a},$$

i.e., Snap randomly perturbs α and then rounds it to a bits of precision. (Here we are borrowing notation from the author's work with Umans [HU21].) We extend the definition to matrices M entrywise, i.e., $\mathsf{Snap}(M, z)_{u,v} = \mathsf{Snap}(M_{u,v}, z)$.

Now we can approximately compute M^w by alternately applying Pow and Snap. That is, let $t = \frac{\log w}{\log n}$, and for simplicity assume that t is an integer. Sample $X \in \{0,1\}^\ell$ and $Z_1, \dots, Z_t \in [2^a]$ independently and uniformly at random, and define a sequence of matrices $\widehat{M}_0, \dots, \widehat{M}_t \in \mathbb{R}^{w^2 \times w^2}$ (random variables) as follows. We start by letting $\widehat{M}_0 \in \{0, \frac{1}{2}, 1\}^{w^2 \times w^2}$ be the transition probability matrix of the directed graph Wrap_f (see Section 2.4). Then we set

$$\widehat{M}_{i+1} = \mathsf{Snap}\left(\mathsf{Pow}\left(\widehat{M}_i, X\right), Z_i\right).$$

The output of our randomized algorithm is given by $E = \left(\widehat{M}_t\right)_{v_{\mathrm{start}}, v_{\mathrm{acc}}}$.

Correctness

For the sake of analysis, define a sequence of matrices $\overline{M}_0, \ldots, \overline{M}_t \in \mathbb{R}^{w^2 \times w^2}$ that depend only on $Z_1, \ldots, Z_t$ (not on X) as follows. We start with $\overline{M}_0 = \widehat{M}_0$. Then we set

$$\overline{M}_{i+1} = \mathsf{Snap}\left(\left(\overline{M}_i\right)^n, Z_i\right).$$

That is, $\overline{M}_i$ is defined like $\widehat{M}_i$, except the approximate powering operation Pow has been replaced by true powering. Note that each $\overline{M}_i$ is a substochastic matrix with a bits of precision.

The following lemma says that Pow approximates matrix powers. For a matrix M, define $\|M\|_{\max} = \max_{u,v} |M_{u,v}|$.

Lemma 4.2.4. *For each $i < t$, for any fixing of $Z_1, \ldots, Z_{i-1}$, with probability $1 - 0.1/w$ over X,*

$$\left\| \left(\overline{M}_i\right)^n - \mathsf{Pow}(\overline{M}_i, X) \right\|_{\max} \leq 3\varepsilon.$$

Proof. For any width-w length-n ROBP g over the alphabet $\{0,1\}^a$, we can define

$$h(e) = \rho(e) \cdot g(G(e)),$$

so h maps $\{0,1\}^s \to [-K, K]$. By the sampler condition, with probability $1 - 0.1/w^5$ over X, we have

$$\left| \mathbb{E}[h] - 2^{-q} \cdot \sum_{y \in \{0,1\}^q} h(\mathsf{Samp}(X, y)) \right| \leq 2\varepsilon.$$

Meanwhile, by the WPRG condition, $|\mathbb{E}[h] - \mathbb{E}[g]| \leq \varepsilon$. Therefore, with probability $1 - 0.1/w^5$

over X, we have

$$\left| \mathbb{E}[g] - 2^{-q} \cdot \sum_{y \in \{0,1\}^q} \rho(\mathsf{Samp}(X, y)) \cdot g(G(\mathsf{Samp}(X, y)) \right| \leq 3\varepsilon.$$

Looking at the ROBP $g^{(\overline{M}_i)}$ in the definition of Pow and applying the union bound over the w^4 entries of the matrix completes the proof. □

The next lemma says that Snap tends to map nearby inputs to the exact same output.

Lemma 4.2.5. *Let $\alpha, \beta \in \mathbb{R}$ and pick $Z \in [2^a]$ uniformly at random. Then*

$$\Pr[\mathsf{Snap}(\alpha, Z) \neq \mathsf{Snap}(\beta, Z)] \leq 2^a \cdot |\alpha - \beta| + 2^{-a}.$$

Proof. Without loss of generality, assume $\beta < \alpha$. The event $\mathsf{Snap}(\alpha, Z) \neq \mathsf{Snap}(\beta, Z)$ occurs if and only if there is some integer Q such that $\beta \cdot 2^a - Z \cdot 2^{-a} < Q \leq \alpha \cdot 2^a - Z \cdot 2^{-a}$. Equivalently, the event occurs if and only if the interval $(\beta \cdot 2^{2a}, \alpha \cdot 2^{2a}]$ contains some integer that is congruent to Z mod 2^a. Now, the number of integers in the interval $(\beta \cdot 2^{2a}, \alpha \cdot 2^{2a}]$ is at most $(\alpha - \beta) \cdot 2^{2a} + 1$. By the union bound, the probability that Z is congruent to one of them mod 2^a is at most $(\alpha - \beta) \cdot 2^a + 2^{-a}$. □

Corollary 4.2.6. *For each $i < t$, for any fixing of $Z_1, \ldots, Z_{i-1}$, for any matrix $M \in \mathbb{R}^{w^2 \times w^2}$,*

$$\Pr_{Z_i}\left[\overline{M}_{i+1} \neq \mathsf{Snap}(M, Z_i)\right] \leq w^4 \cdot \left(2^a \cdot \left\| M - \left(\overline{M}_i\right)^n \right\|_{\max} + 2^{-a}\right).$$

Proof. Apply Lemma 4.2.5 to each entry and apply the union bound. □

The magical consequence is that after applying the Snap operation, it is as if we took a true power instead of taking the approximate power using Pow.

Corollary 4.2.7. *With probability at least* $2/3$*, for all* i *simultaneously,* $\widehat{M_i} = \overline{M}_i$.

Proof. By Lemma 4.2.4, Corollary 4.2.6, and the union bound, the failure probability is at most

$$0.1 \cdot t/w + t \cdot w^4 \cdot (2^a \cdot 3\varepsilon + 2^{-a}).$$

Recall that $t \leq w$, $2^a \in [4w^7, 8w^7]$, and $\varepsilon = w^{-12}/24$. Therefore, the failure probability is at most $1/3$ as claimed. □

Next, we need to bound the total amount of error introduced by all the applications of the Snap operation. For a matrix M, let $\|M\|_\infty$ denote the maximum sum of absolute values of entries in any row of M. This norm has the following convenient (and standard) property.

Lemma 4.2.8. *If* A *and* B *are substochastic and* $n \in \mathbb{N}$*, then* $\|A^n - B^n\|_\infty \leq n \cdot \|A - B\|_\infty$.

Proof. If A' is substochastic and B' is any matrix, then

$$\begin{aligned}
\|AA' - BB'\|_\infty &\leq \|AA' - BA'\|_\infty + \|BA' - BB'\|_\infty \\
&\leq \|A - B\|_\infty \cdot \|A'\|_\infty + \|B\|_\infty \cdot \|A' - B'\|_\infty \qquad \text{(Submultiplicativity)} \\
&\leq \|A - B\|_\infty + \|A' - B'\|_\infty.
\end{aligned}$$

The lemma follows by induction on n. □

Now we are ready to bound the error introduced by Snap.

Lemma 4.2.9. *With probability* 1*, we have* $\|\overline{M}_t - (\overline{M}_0)^w\|_{\max} \leq 4w^3 \cdot 2^{-a}$.

Proof. For any $\alpha \in \mathbb{R}$ and any $z \in [2^a]$, clearly $|\alpha - \mathsf{Snap}(\alpha, z)| \le 2 \cdot 2^{-a}$. Therefore, for each $i < t$,

$$\|\overline{M}_{i+1} - (\overline{M}_i)^n\|_{\max} \le 2 \cdot 2^{-a}.$$

As a consequence,

$$\begin{aligned}
\|\overline{M}_{i+1} - (\overline{M}_0)^{n^{i+1}}\|_\infty &\le \|\overline{M}_{i+1} - (\overline{M}_i)^n\|_\infty + \|(\overline{M}_i)^n - (\overline{M}_0)^{n^{i+1}}\|_\infty \\
&\le w^2 \cdot \|\overline{M}_{i+1} - (\overline{M}_i)^n\|_{\max} + n \cdot \|\overline{M}_i - (\overline{M}_0)^{n^i}\|_\infty \\
&\le 2w^2 \cdot 2^{-a} + n \cdot \|\overline{M}_i - (\overline{M}_0)^{n^i}\|_\infty.
\end{aligned}$$

By induction, it follows that

$$\|\overline{M}_t - (\overline{M}_0)^{n^t}\|_\infty \le 4w^2 \cdot 2^{-a} \cdot n^t = 4w^3 \cdot 2^{-a}. \qquad \square$$

Corollary 4.2.10. $\Pr[|E - \mathbb{E}[f]| \le 1/w^4] \ge 2/3$.

Proof. This follows from Corollary 4.2.7, Lemma 4.2.9, and our choice $a = \lceil 2 + 7 \log w \rceil$. $\square$

Efficiency

We will take Samp to be the sampler of Theorem 4.1.12, so

$$\begin{aligned}
\ell &= s + O(\log(Kw/\varepsilon)) = s + O(\log(Kw)) \\
q &= O(\log(K/\varepsilon) + \log\log w) = O(\log(Kw)).
\end{aligned}$$

Clearly, the number of random bits involved in the computation of E is $\ell + ta$, which is

$$s + O\left(\log(Kw) + \frac{\log^2 w}{\log n}\right) = O\left(\frac{\log^{3/2} w}{\sqrt{\log\log w}}\right)$$

as desired. Now we will show that the computation of E is space-efficient. We will apply Lemma 2.1.1. To do so, define $F(M,(x,z)) = \mathsf{Snap}(\mathsf{Pow}(M,x),z)$. We wish to compute F in $O(s+\log(Kw))$ space in such a way that whenever we read from M, we first delete all but $O(\log(Kw))$ bits of the work space.

Indeed, the straightforward algorithm for F has that property. Computing Snap uses just $O(a+\log w) = O(\log w)$ bits of space. To compute $\mathsf{Pow}(M,x)_{u,v}$, we must iterate over all $y \in \{0,1\}^q$, compute $G(\mathsf{Samp}(x,y))$ and feed it into $g^{(M)}_{(u,0)\to(v,n)}$, compute $\rho(\mathsf{Samp}(x,y))$, and keep a running sum. The string y can be stored using $q = O(\log(Kw))$ bits of space. The sampler Samp runs in space $O(s+\log(Kw/\varepsilon)) = O(s+\log(Kw))$, and G and ρ both run in space $O(s)$. The transitions of $g^{(M)}$ can be computed in $O(a+\log w) = O(\log w)$ bits of space, and since ρ takes on integer values in $[-K,K]$, the running sum can be stored using $O(q+\log K) = O(\log(Kw))$ bits of space. The key thing is that when we read from M, we are not in the middle of computing Samp or G or ρ; we are just storing a single a-bit symbol from the output of G. Thus, whenever we read from M, we first delete all but $O(a+\log(Kw)) = O(\log(Kw))$ bits of the work space.

Now, in the notation of Lemma 2.1.1, the matrix $\widehat{M_t}$ is given by

$$F^{(t)}(\widehat{M_0},(X,Z_1),\dots,(X,Z_t)).$$

Thus, to compute $\widehat{M_t}$, we can pick $X,Z_1,\dots,Z_t$ at random and store them on the work space, then apply the algorithm of Lemma 2.1.1, which runs in space $O(s+t\log(Kw))$. Counting also the space for storing $X,Z_1,\dots,Z_t$, the total space complexity is

$$O(s+t\log(Kw)) = O\left(s+\frac{\log(Kw)\log w}{\log n}\right) = O\left(\frac{\log^{3/2} w}{\sqrt{\log\log w}}\right)$$

as desired. This completes the proof of Lemma 4.2.3.

Theorem 4.2.11 (Theorem 3.2.3 restated)**.** *Given a width-w length-w ROBP f, it is possible to estimate $\mathbb{E}[f]$ to within $\pm 1/w^4$ deterministically in space $O\left(\log^{3/2} w/\sqrt{\log\log w}\right)$.*

Proof. Run the algorithm of Lemma 4.2.3 on all possible settings of its random bits and take the median output. □

Now we turn to the derandomization of **BPSPACE**(S) in space $o(S^{3/2})$ (Theorem 4.2.1). When S is space-constructible, the theorem follows from Theorem 4.2.11, Proposition 2.4.1, and a padding argument. When S is not space-constructible, the theorem still holds; the idea is to try larger and larger space bounds until eventually finding a value $\widetilde{S}$ such that the randomized algorithm has at most a small probability of using more than $\widetilde{S}$ bits of space. We omit the details.

Note that in general, if we had started with a WPRG like the one given by Lemma 4.2.2 but with potentially different values of n, s, and K, then the space complexity in Theorem 4.2.11 would have been

$$O\left(s + \frac{\log(Kw)\log w}{\log n}\right).$$

Chapter 5

PRG Constructions and Lower Bounds

In this chapter,[1] we study PRGs (the traditional, unweighted version). We are unfortunately still not able to obtain improved PRGs for unrestricted ROBPs, but we will present improved PRGs for several related models of computation. In Section 5.1, we present a PRG for unbounded-width permutation ROBPs, along with a near-matching (non-trivial) lower bound on the seed length required to fool such programs. Then, in Section 5.2, we present a

[1] For clarity, we provide here the full citations for the papers that this chapter is based on.

- [HPV21] William M. Hoza, Edward Pyne, and Salil Vadhan. "Pseudorandom Generators for Unbounded-Width Permutation Branching Programs". In: *Proceedings of the 12th Innovations in Theoretical Computer Science Conference (ITCS)*. 2021, 7:1–7:20. ISBN: 978-3-95977-177-1. DOI: 10.4230/LIPIcs.ITCS.2021.7
- [DHH19] Dean Doron, Pooya Hatami, and William M. Hoza. "Near-Optimal Pseudorandom Generators for Constant-Depth Read-Once Formulas". In: *Proceedings of the 34th Computational Complexity Conference (CCC)*. 2019, 16:1–16:34. DOI: 10.4230/LIPIcs.CCC.2019.16
- [HHTT21] Pooya Hatami, William M. Hoza, Avishay Tal, and Roei Tell. *Fooling Constant-Depth Threshold Circuits*. 2021. ECCC: TR21-002

As usual, each paper is a collaborative effort, and in each case, each author contributed both to the intellectual development of the ideas in the paper and to the writing process.

near-optimal PRG for read-once $\mathbf{AC}^0$. Finally, in Section 5.3, we present a PRG for threshold circuits that essentially matches the state-of-the-art lower bounds for this powerful model of computation. Each PRG has some connection to the derandomization of space-bounded computation, as we will explain.

The first PRG is joint work with Ted Pyne and Salil Vadhan [HPV21]. A video presentation by Ted is available online [Pyn21]. The second PRG is joint work with Dean Doron and Pooya Hatami [DHH19], and the last PRG is joint work with Pooya Hatami, Avishay Tal, and Roei Tell [HHTT21]. Videos of presentations of these two PRGs by the author are available online [Hoz19a; Hoz21a].

5.1 PRGs for Unbounded-Width Permutation ROBPs

To make progress on fooling ROBPs, researchers have looked at restricted classes of ROBPs. One well-studied class is *permutation* ROBPs [BV10; De11; KNP11; Ste12; RSV13; CHHL19; HPV21; PV21].

Definition 5.1.1. *A width-w length-n ROBP f is a* permutation ROBP *if for each $i \in [n]$ and each $b \in \{0,1\}$, the transition function $\mathrm{next}_i(\cdot, b)$ is a permutation on the state space $[w]$. Equivalently, there is no pair of edges (u, v) and (u', v) with the same label where $u \neq u'$.*

The computation performed by a permutation ROBP is *reversible* in a certain sense: we can deduce the previous state of the computation by looking at the current state and the input bit that was most recently read. Permutation ROBPs are a special case of regular ROBPs. We say that an ROBP is *regular* if every vertex not in the last layer has in-degree 2, or equivalently, Wrap_f is a regular digraph.

Most of the work on regular and permutation ROBPs has focused on the constant-width setting [BV10; De11; KNP11; Ste12; RSV13; BRRY14; CHHL19]. This body of work

played a key role in Meka, Reingold, and Tal's PRG for width-3 ROBPs [MRT19].

In this section, we are interested in PRGs for wider permutation ROBPs. The polynomial-width case is arguably the most important, since unrestricted polynomial-width ROBPs model **BPL** (Proposition 2.4.1). Furthermore, there is a reduction due to Reingold, Trevisan, and Vadhan showing that optimal PRGs for polynomial-width regular ROBPs would actually suffice for derandomizing **RL** [RTV06]. We have seen that the Saks-Zhou algorithm [SZ99] involves a PRG for ROBPs of *superpolynomial* width, $w \approx n^{\log n}$, which gives us extra motivation for studying very wide programs.

We will skip all the way to *unbounded-width* permutation ROBPs. Crucially, like in Definition 1.2.2, we only allow a single accept vertex v_{acc}. (For unrestricted ROBPs, it is no loss of generality to assume that there is only one accept vertex v_{acc}, because multiple accept vertices could always be merged. For permutation ROBPs, however, it makes a big difference. Unbounded-width permutation ROBPs with an unbounded number of accept vertices are all-powerful, i.e., they can compute any Boolean function, whereas one can show that unbounded-width permutation ROBPs with a single accept vertex cannot compute MAJ_3.)

This unusual model of computation reveals some limits of standard intuitions about pseudorandomness for branching programs. We will see (Lemma 5.1.20) that these programs can compute doubly-exponentially many functions, so the generic probabilistic argument for the existence of PRGs (Proposition 1.2.3) does not apply. In fact, it turns out that with high probability, a random function with seed length $o(n)$ *fails* to be a good PRG for this model [HPV21]. Nevertheless, in Section 5.1.1, we will show that there is an explicit PRG (the classic INW generator [INW94]) that ε-fools unbounded-width length-n permutation ROBPs with seed length $\widetilde{O}(\log n \cdot \log(1/\varepsilon))$. Then in Section 5.1.2, we will prove a near-matching lower bound, i.e., every ε-PRG for unbounded-width length-n permutation ROBPs must have seed length $\widetilde{\Omega}(\log n \cdot \log(1/\varepsilon))$. Both of these results are joint work with Ted Pyne

and Salil Vadhan [HPV21].

5.1.1 Improved Analysis of the INW Generator

In this section, we present an explicit PRG for permutation ROBPs with seed length $\widetilde{O}(\log n \cdot \log(1/\varepsilon))$, with no dependence whatsoever on the width of the program.

Theorem 5.1.2 (Joint with Ted Pyne and Salil Vadhan [HPV21])**.** *For every $n \in \mathbb{N}$ and $\varepsilon > 0$, there is an explicit ε-PRG for length-n permutation ROBPs (of any width) with seed length $O(\log n \cdot (\log(1/\varepsilon) + \log\log n))$.*

We reiterate that our model of permutation ROBP only allows a single accepting vertex. That being said, a permutation ROBP with $a \geq 1$ accepting vertices can be written as a sum of a permutation ROBPs with one accepting vertex each, so our PRG ε-fools such programs with seed length $O(\log n \cdot (\log(a/\varepsilon) + \log\log n))$.

As mentioned previously, most prior work on permutation ROBPs focused on constant-width programs. In that regime, there are known PRGs with seed length $O(\log n \cdot \log(1/\varepsilon))$ [De11; KNP11; Ste12], which is slightly better than our seed length. For somewhat larger width, the best prior PRG is Braverman, Rao, Raz, and Yehudayoff's generator [BRRY14], which fools width-w length-n regular ROBPs with seed length

$$O(\log n \cdot (\log(w/\varepsilon) + \log\log n)).$$

For superpolynomial-width ROBPs, the best prior PRG is by De; his paper [De11] was focused on the constant-width case, but he also showed that the INW generator fools unbounded-width permutation ROBPs with seed length $O(\log^2 n + \log n \cdot \log(1/\varepsilon)$ (see Table 5.1).

Theorem 5.1.2 should also be compared to the work of Ahmadinejad, Kelner, Murtagh,

Seed length	Type of ROBPs	Reference
$O(\log n \cdot (\log(1/\varepsilon) + (w!)^{11}))$	Permutation	[KNP11]
$O(\log n \cdot (\log(1/\varepsilon) + w^8))$	Permutation	[De11]
$O(\log n \cdot \log(n/\varepsilon))$	Permutation	[De11]
$O(\log n \cdot (\log(1/\varepsilon) + w^4 \log w))$	Permutation	[Ste12]
$O(\log n \cdot (\log(w/\varepsilon) + \log\log n))$	Regular	[BRRY14]
$O(\log n \cdot (\log(1/\varepsilon) + \log\log n)$	Permutation	[HPV21] (Theorem 5.1.2)

Table 5.1: Known PRGs for width-w length-n permutation ROBPs. For comparison, recall that Nisan's PRG for unrestricted ROBPs [Nis92] has seed length $O(\log n \cdot \log(wn/\varepsilon))$. This table omits known PRGs for the more challenging model of *arbitrary-order* permutation ROBPs [RSV13; CHHL19].

Peebles, Sidford, and Vadhan [AKMPSV20]. Among other results, they give a non-black-box algorithm for estimating the acceptance probability of a given regular width-n length-n ROBP to within $\pm 1/\operatorname{poly}(n)$ in space $O(\log n \cdot \log\log(n/\varepsilon))$ [AKMPSV20]. We only handle permutation ROBPs and we have a much worse dependence on ε, but we obtain a genuine PRG (and as we will see, our dependence on ε is near-optimal for PRGs). Finally, subsequent to and building on our work, Pyne and Vadhan recently designed a WPRG for unbounded-width permutation ROBPs [PV21] with seed length

$$\widetilde{O}\left(\log^{3/2} n + \log n \cdot \sqrt{\log(1/\varepsilon)} + \log(1/\varepsilon)\right),$$

which is better than our seed length when ε is small.

The construction

The construction of Theorem 5.1.2 is the standard INW generator [INW94] with suitable parameters. The analysis is what is new – although as we shall see, even the analysis follows without too much difficulty from the prior works by Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan [AKMPSV20] and Rozenman and Vadhan [RV05]. We begin by reviewing the INW construction, which is based on *expanders.*

Definition 5.1.3. *Let H be an undirected regular multigraph on N vertices, let M be the random walk matrix of H, and let $\lambda \in (0,1)$. We say that H is a* λ-spectral expander *if for every vector $x \in \mathbb{R}^N$ that is orthogonal to the all-ones vector, we have $\|Mx\|_2 \leq \lambda \|x\|_2$.*

For a value $\lambda = \Theta(\frac{\varepsilon}{\log n})$ to be chosen later, let $H_1, H_2, \ldots$ be a sequence of λ-spectral expanders, all of the same degree $D = 2^d$, where H_i is on D^i vertices. We require that each such graph has a *one-way labeling*:

Definition 5.1.4. *Let H be a D-regular directed multigraph. A* one-way labeling *of H assigns a distinct label in $[D]$ to each of the D outgoing edges of each vertex. For a vertex v and a label $e \in [D]$, we define $H[v,e]$ to be the outneighbor of v reached by traversing the outgoing edge labeled e.*

There exists an explicit such family of expanders $H_1, H_2, \ldots$ with $d = O(\log(1/\lambda))$ [Vad12]. We define a family of generators $\mathrm{INW}_i \colon \{0,1\}^{d\cdot(i+1)} \to \{0,1\}^{d\cdot 2^i}$ by

$$\begin{aligned} \mathrm{INW}_0(x) &= x \\ \mathrm{INW}_{i+1}(x,y) &= \mathrm{INW}_i(x) +\!\!+ \mathrm{INW}_i(H_{i+1}[x,y]), \end{aligned}$$

where $+\!\!+$ indicates concatenation of strings. The PRG that will prove Theorem 5.1.2 is $\mathrm{INW} \stackrel{\text{def}}{=} \mathrm{INW}_{\log(n/d)}$.

Derandomized square

The first part of the analysis is to review the connection between the INW generator and *derandomized squaring*. The derandomized square operation was introduced by Rozenman and Vadhan [RV05], who used it to give an alternative proof of Reingold's famous theorem that undirected connectivity is in **L** [Rei08]. More generally, they showed how to generate polynomial-length pseudorandom walks through regular, aperiodic, directed graphs such that the distribution of the final vertex in the walk is approximately uniform [RV05], giving an alternative proof of a result of Reingold, Trevisan, and Vadhan [RTV06]. Most relevant for us, they showed that in consistently labeled graphs (see Definition 5.1.8), this pseudorandom walk generator is equivalent to the INW generator [INW94].

A more recent line of work has shown how to approximate *short* random walks (i.e., walks of length much less than the mixing time of the graph) in undirected graphs and Eulerian digraphs [MRSV17; MRSV19; AKMPSV20]. These recent works use the derandomized square operation in combination with other tools. We will build on their analyses, especially the analysis by Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan [AKMPSV20].

Toward developing the derandomized square, we begin by defining the *true* square of a graph, and more generally the n-th power of a graph.

Definition 5.1.5. *Let G be a D-regular directed multigraph with a one-way labeling on vertex set $[N]$. If n is a positive integer, then G^n is a D^n-regular directed multigraph with a one-way labeling on vertex set $[N]$ given recursively by*

$$G^1[u, e] = G[u, e]$$
$$G^{n+1}[u, (e, e')] = G^n[G[u, e], e'],$$

identifying $[D^{n+1}]$ with $[D] \times [D^n]$.

To apply the derandomized squaring operation, we will require a graph equipped with a *two-way* labeling.

Definition 5.1.6. *Let G be a D-regular directed multigraph. A* two-way labeling *of G assigns two labels to each edge, an "incoming label" and an "outgoing label." Each label is an element of $[D]$, and we require that for each vertex v, the outgoing labels of the outgoing edges are distinct, and the incoming labels of the incoming edges are distinct. A two-way labeling induces a one-way labeling by considering only the outgoing labels. We define the* rotation map $\mathrm{Rot}_G\colon V(G) \times [D] \to V(G) \times [D]$ *by letting* $\mathrm{Rot}(u, e) = (G[u, e], e')$*, where e' is the incoming label of the edge $(u, G[u, e])$.*

The derandomized square $G \circledS H$ is a method of approximating G^2 by a sparser graph. In our analysis, the auxiliary graph H will be an expander.

Definition 5.1.7. *Let G be a D-regular directed multigraph with a two-way labeling on vertex set $[N]$, and let H be a c-regular directed multigraph with a one-way labeling on vertex set $[D]$. The* derandomized square $G \circledS H$ *is a (cD)-regular directed multigraph on vertex set $[N]$ with a one-way labeling given by*

$$(G \circledS H)[v, (i, j)] = G[v', H[i', j]],$$

where $(v', i') = \mathrm{Rot}_G(v, i)$.

Notice that in the definition of $\circledS$, we require that G has a two-way labeling, but we merely define a one-way labeling for $G \circledS H$. To iterate the process and approximate G^4, we must somehow obtain a two-way labeling for $G \circledS H$. Multiple approaches have been studied in the literature [RV05]; the relevant approach for us is to assume a *consistent one-way labeling*.

Definition 5.1.8. *Let G be a D-regular directed multigraph. A* consistent one-way labeling *of G is a one-way labeling of G with the property that for every vertex v, the labels of the incoming edges are distinct. Equivalently, for every vertex v and every $e \in [D]$, there is a vertex u such that $G[u, e] = v$. Such a labeling allows us to extend G to a graph $\overline{G}$ with a two-labeling given by*

$$\mathrm{Rot}_{\overline{G}}(v, e) = (G[v, e], e),$$

i.e., each edge has the same incoming label and outgoing label.

The property of having a consistent one-way labeling is obviously preserved under powering. It is also preserved by the derandomized square, so indeed, it enables iterative derandomized squaring:

Lemma 5.1.9 ([RV05; HPV21])**.** *If G has a consistent one-way labeling, then so does $\overline{G} Ⓢ H$.*

Proof. Fix a vertex v and an edge label (i, j); we must show that there is some u such that $(\overline{G} Ⓢ H)[u, (i, j)] = v$. Indeed, since G has a consistent one-way labeling, there is some u' such that $G[u', H[i, j]] = v$; again using the fact that G has a consistent labeling, there is some u such that $G[u, i] = u'$. Finally, $(\overline{G} Ⓢ H)[u, (i, j)] = G[G[u, i], H[i, j]] = v$. □

For graphs with a consistent one-way labeling, the INW generator corresponds closely to the derandomized square as follows.

Lemma 5.1.10 ([RV05; HPV21])**.** *Let G_0 be a consistently labeled D-regular multigraph. Inductively define $G_1, G_2, \ldots$ by $G_i = \overline{G_{i-1}} Ⓢ H_i$. Then*

$$G_i[v, e] = G_0^{2^i}[v, \mathrm{INW}_i(e)].$$

Proof. In the base case $i = 0$, this is trivial. For the inductive step $i > 0$, let $G = G_0^{2^{i-1}}$.

Then

$$\begin{aligned} G_0^{2^i}[v, \mathrm{INW}_i(x,y)] &= G^2[v, \mathrm{INW}_i(x,y)] \\ &= G[G[v, \mathrm{INW}_{i-1}(x)], \mathrm{INW}_{i-1}(H_i[x,y])] \\ &= G_{i-1}[G_{i-1}[v,(x,y)], H_i[x,y]] && \text{(Induction)} \\ &= (\overline{G_{i-1}} \circledS H_i)[v,(x,y)] \\ &= G_i[v,(x,y)]. \end{aligned}$$

□

In our case, we are interested in analyzing the behavior of a permutation ROBP f when it reads the output of the INW generator. Recall that for any width-w length-n ROBP f, there is a corresponding 2-outregular digraph Wrap_f on vertex set $[w] \times [n]$ obtained from f by identifying each vertex in the last layer with the corresponding vertex in the first layer. The graph Wrap_f has a one-way labeling, and for any $x \in \{0,1\}^n$, we have

$$f(x) = 1 \iff \mathrm{Wrap}_f^n[v_{\mathrm{start}}, x] = v_{\mathrm{acc}}.$$

When f is a permutation ROBP, Wrap_f has a *consistent* one-way labeling, which allows us to define a sequence of graphs $G_0, G_1, \ldots$ by $G_0 = \mathrm{Wrap}_f^d$ and $G_{i+1} = \overline{G_i} \circledS H_{i+1}$. We have

$$f(x) = 1 \iff G_0^{n/d}[v_{\mathrm{start}}, x] = v_{\mathrm{acc}}, \tag{5.1}$$

and for any seed e,

$$\begin{aligned} f(\mathrm{INW}(e)) = 1 &\iff G_0^{n/d}[v_{\mathrm{start}}, \mathrm{INW}(e)] = v_{\mathrm{acc}} && \text{(Eq. (5.1))} && (5.2) \\ &\iff G_{\log(n/d)}[v, e] = v_{\mathrm{acc}} && \text{(Lemma 5.1.10).} && (5.3) \end{aligned}$$

Thus, comparing the acceptance probabilities of f on a truly random input vs. the output of the INW generator is equivalent to comparing the random walk matrix of the true power $G_0^{n/d}$ vs. the random walk matrix of the iterated derandomized square $G_{\log(n/d)}$. We will compare those matrices next.

Unit-circle approximation

Our goal is to show that the random walk matrices of G_i and $G_0^{2^i}$ are $O(i\lambda)$-close entrywise. The key to proving this bound is to consider more sophisticated notions of matrix approximation. Even though all our matrices are real-valued, it will be helpful to move to the field of complex numbers.

For a matrix $M \in \mathbb{C}^{N\times N}$, we define the *symmetrization* $U_M = \frac{1}{2}(M + M^*)$, where M^* is the conjugate transpose of M. Our first notion of matrix approximation is *complex spectral approximation*.

Definition 5.1.11 ([AKMPSV20])**.** *Let $M, \widetilde{M} \in \mathbb{C}^{N\times N}$ and $\gamma > 0$. We say that $\widetilde{M}$ is a* complex spectral γ-approximation *of M, denoted $\widetilde{M} \approx_\gamma M$, if*

$$\forall x, y \in \mathbb{C}^N, \quad |x^*(M - \widetilde{M})y| \leq \frac{\gamma}{2} \cdot (\|x\|^2 + \|y\|^2 - x^* U_M x - y^* U_M y). \tag{5.4}$$

The analogous approximation measure over the reals was studied by Cohen, Kelner, Peebles, Peng, Rao, Sidford, and Vladu [CKPPRSV17], building on earlier work by Spielman and Teng [ST11]. For intuition, note that when M and $\widetilde{M}$ are real symmetric matrices, if we let $L = I - M$ and $\widetilde{L} = I - \widetilde{M}$, then Eq. (5.4) implies that for every $x \in \mathbb{R}^N$, we have

$$(1-\gamma)x^T L x \leq x^T \widetilde{L} x \leq (1+\gamma)x^T L x.$$

In other words, $(1-\gamma)L \preceq \widetilde{L} \preceq (1+\gamma)L$, where $\preceq$ denotes Loewner order. Observe that if

$Mx = x$, then $\widetilde{M}x = x$ as well.

In the general case, we allow $x \neq y$ in Eq. (5.4) to measure whatever "asymmetric information" is contained in the matrices M and $\widetilde{M}$. Still, complex spectral approximation has certain weaknesses. For example, it is not preserved under graph powering [AKMPSV20, Proposition 4.3]. The issue is that complex spectral approximation allows for the possibility that $Mx = \omega x$, where ω is a k-th *root* of unity, while $\widetilde{M}x = (1-\gamma)\omega x$. When we take k-th powers, we get $M^k x = x$ but $\widetilde{M}^k x \neq x$, so $\widetilde{M}^k \not\approx M^k$.

Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan therefore introduced a stronger notion of matrix approximation, called *unit circle approximation* [AKMPSV20]. Rather than the original definition, we will work with an equivalent condition [AKMPSV20, Lemma 3.8].

Definition 5.1.12 ([AKMPSV20])**.** *Let $M, \widetilde{M} \in \mathbb{C}^{N\times N}$ and $\gamma > 0$. We say that $\widetilde{M}$ is a* unit circle γ-approximation *of M, denoted $\widetilde{M} \overset{\circ}{\approx}_\gamma M$, if for every $z \in \mathbb{C}$ with $|z| = 1$, we have $z\widetilde{M} \approx_\gamma zM$. If M and $\widetilde{M}$ are the random walk matrices of digraphs G and $\widetilde{G}$ respectively, then we also write $\widetilde{G} \overset{\circ}{\approx}_\gamma G$.*

Unit circle approximation implies that M and $\widetilde{M}$ must exactly agree on every eigenvector x of M with an eigenvalue on the complex unit circle, not just those with eigenvalue 1. This gives us hope that $\widetilde{G} \overset{\circ}{\approx} G$ implies $\widetilde{G}^k \overset{\circ}{\approx} G^k$. Intuitively, it seems like at best, the error should blow up by perhaps a factor of k when we take k-th powers. After all, a single step in $\widetilde{G}^k$ consists of k steps in $\widetilde{G}$, each of which is slightly erroneous. Contrary to this intuition, Ahmadinejad et al. showed that remarkably, graph powering preserves unit circle approximation with essentially no loss:

Lemma 5.1.13 ([AKMPSV20, Corollary 4.9])**.** *Let $\widetilde{G}$ and G be directed multigraphs with $\widetilde{G} \overset{\circ}{\approx}_\gamma G$. Then for all $k \in \mathbb{N}$, $\widetilde{G}^k \overset{\circ}{\approx}_{\gamma'} G^k$, where $\gamma' = \frac{\gamma}{1-3\gamma/2}$.*

Unit-circle approximation and iterated derandomized squaring

Recall that we have a sequence of graphs $G_0, G_1, \ldots$ where $G_0 = \mathrm{Wrap}_f^d$ and $G_{i+1} = \overline{G_i} \circledS H_{i+1}$. To show that G_i and $G_0^{2^i}$ are close entrywise, we will prove the stronger statement that $G_i \mathrel{\mathring{\approx}} G_0^{2^i}$:

Claim 5.1.14. *If $i\lambda \leq 1/10$[2], then $G_i \mathrel{\mathring{\approx}}_{4i\lambda} G_0^{2^i}$.*

The case $i = 1$ was proven already by Ahmadinejad et al. [AKMPSV20].

Lemma 5.1.15 ([AKMPSV20, Theorem 5.9])**.** *If G is a regular directed multigraph with a two-way labeling and H is a λ-spectral expander, then $G \circledS H \mathrel{\mathring{\approx}}_{2\lambda} G^2$.*

To analyze larger powers, we rely on an approximate triangle inequality for unit-circle approximation.

Lemma 5.1.16 (Quasi-Triangle Inequality)**.** *Let $M_1, M_2, M_3 \in \mathbb{C}^{N\times N}$. If $M_1 \mathrel{\mathring{\approx}}_{\gamma_1} M_2 \mathrel{\mathring{\approx}}_{\gamma_2} M_3$, then*

$$M_1 \mathrel{\mathring{\approx}}_{\gamma_1+\gamma_2+\gamma_1\gamma_2} M_3.$$

Proof. Let $z \in \mathbb{C}$ with $|z| = 1$ and let $x, y \in \mathbb{C}^N$. Since $zM_1 \approx_{\gamma_1} zM_2 \approx_{\gamma_2} zM_3$, we have

$$|x^*(M_2 - M_1)y| \leq \frac{\gamma_1}{2}(x^*U_{I-zM_2}x + y^*U_{I-zM_2}y) \tag{5.5}$$

$$|x^*(M_3 - M_2)y| \leq \frac{\gamma_2}{2}(x^*U_{I-zM_3}x + y^*U_{I-zM_3}y). \tag{5.6}$$

On the right-hand side of Eq. (5.5), we would like to replace M_2 by M_3. To bound the effect

[2]This is stronger than the assumption in the paper [HPV21]. The proof in the paper is not correct, due to a mistake in the "iterated quasi-triangle inequality" [HPV21, Corollary 2.6]. The corrected version of the iterated quasi-triangle inequality is given in Corollary 5.1.17.

of such a substitution, note that

$$\begin{aligned}|x^* U_{I-zM_3} x - x^* U_{I-zM_2} x| &= \left| \frac{x^*(zM_3 + (zM_3)^*)x - x^*(zM_2 + (zM_2)^*)x}{2} \right| \\ &= |x^*(M_3 - M_2)x| \\ &\leq \gamma_2 \cdot x^* U_{I-zM_3} x,\end{aligned}$$

where the last inequality is by plugging $y = x$ into Eq. (5.6). Similarly,

$$|y^* U_{I-zM_3} y - y^* U_{I-zM_2} y| \leq \gamma_2 \cdot y^* U_{I-zM_3} y.$$

Therefore,

$$|x^*(M_2 - M_1)y| \leq \frac{\gamma_1}{2} \cdot (1 + \gamma_2) \cdot (x^* U_{I-zM_3} x + y^* U_{I-zM_3} y),$$

and thus

$$\begin{aligned}|x^*(zM_3 - zM_1)y| &\leq |x^*(M_3 - M_2)y| + |x^*(M_2 - M_1)y| \\ &\leq \left(\frac{\gamma_2}{2} + \frac{\gamma_1}{2} \cdot (1 + \gamma_2) \right) \cdot (x^* U_{I-zM_3} x + y^* U_{I-zM_3} y),\end{aligned}$$

i.e., $zM_1 \approx_\gamma zM_3$ where $\gamma = \gamma_2 + \gamma_1 \cdot (1 + \gamma_2) = \gamma_1 + \gamma_2 + \gamma_1\gamma_2$. □

Applying Lemma 5.1.16 several times gives the following corollary.

Corollary 5.1.17 (Iterated Quasi-Triangle Inequality)**.** *Suppose* $M_0 \mathrel{\mathring{\approx}}_\delta M_1 \mathrel{\mathring{\approx}}_\delta \cdots \mathrel{\mathring{\approx}}_\delta M_i$*, where* $i\delta < 1$*. Then* $M_0 \mathrel{\mathring{\approx}}_\gamma M_i$*, where* $\gamma = \frac{i\delta}{1-i\delta}$*.*

Proof. Let $\gamma_i = (1 + \delta)^i - 1$. We will show by induction that $M_0 \mathrel{\mathring{\approx}}_{\gamma_i} M_i$. The base case $i = 1$ is trivial. For the inductive step $i > 1$, we have $M_0 \mathrel{\mathring{\approx}}_\delta M_1 \mathrel{\mathring{\approx}}_{\gamma_{i-1}} M_i$, and hence by Lemma 5.1.16,

$$M_0 \mathrel{\mathring{\approx}}_{\delta + \gamma_{i-1} + \delta\gamma_{i-1}} M_i.$$

Furthermore,

$$\delta + \gamma_{i-1} + \delta\gamma_{i-1} = \delta + (1+\delta)\cdot((1+\delta)^{i-1} - 1) = (1+\delta)^i - 1 = \gamma_i,$$

completing the induction. Finally,

$$\gamma_i = (1+\delta)^i - 1 \le e^{i\delta} - 1 \le \frac{1}{1-i\delta} - 1 = \frac{i\delta}{1-i\delta},$$

using the inequalities $1 + t \le e^t$ (valid for all $t \in \mathbb{R}$) and $e^t \le \frac{1}{1-t}$ (valid for all $t < 1$). □

Proof of Claim 5.1.14. By Lemma 5.1.15, for each j, we have $G_{j+1} \mathrel{\mathring{\approx}}_{2\lambda} G_j^2$. By Lemma 5.1.13 with $k = 2^{i-j-1}$, this implies $G_{j+1}^{2^{i-j-1}} \mathrel{\mathring{\approx}}_{\delta} G_j^{2^{i-j}}$, where $\delta = \frac{2\lambda}{1-3\lambda}$. We therefore have a chain of approximations,

$$G_i \mathrel{\mathring{\approx}}_{\delta} G_{i-1}^2 \mathrel{\mathring{\approx}}_{\delta} G_{i-2}^{2^2} \mathrel{\mathring{\approx}}_{\delta} \cdots \mathrel{\mathring{\approx}}_{\delta} G_0^{2^i}.$$

Now, $i\delta = \frac{2i\lambda}{1-3\lambda} < 4i\lambda < 1$, so we may apply Corollary 5.1.17, giving $G_i \mathrel{\mathring{\approx}}_{\gamma} G_0^{2^i}$, where

$$\gamma = \frac{i\delta}{1-i\delta} = \frac{2i\lambda/(1-3\lambda)}{1-2i\lambda/(1-3\lambda)} = \frac{2i\lambda}{1-(2i+3)\lambda} \le 4i\lambda.$$ □

We can now wrap up the proof of correctness of the generator.

Claim 5.1.18. *For a suitable $\lambda = \Theta(\varepsilon/\log n)$, we have $|\mathbb{E}[f(\mathrm{INW}(U_s))] - \mathbb{E}[f]| \le \varepsilon$, where s is the seed length of* INW.

Proof. By Eqs. (5.1) and (5.3), $\mathbb{E}[f]$ is an entry of $G_0^{n/d}$ and $\mathbb{E}[f(\mathrm{INW}(U_s))]$ is the corresponding entry of $G_{\log(n/d)}$. By Claim 5.1.14, $G_0^{n/d} \mathrel{\mathring{\approx}}_{\varepsilon} G_{\log(n/d)}$, which immediately implies that the two matrices are ε-close entrywise (take $z = 1$ and take x and y to be indicator vectors in the definition of unit-circle approximation). □

Seed length

The seed length of INW is

$$O(d \log n) = O(\log n \cdot \log(1/\lambda)) = O(\log n \cdot (\log(1/\varepsilon) + \log\log n))$$

as claimed, completing the proof of Theorem 5.1.2.

5.1.2 Seed Length Lower Bound

In this section, we prove a seed length lower bound that shows that our PRG for unbounded-width permutation ROBPs is near-optimal. For context, it is typically trivial to prove optimal seed length lower bounds for PRGs – but the unbounded-width permutation ROBP is not a typical model of computation!

Theorem 5.1.19 (Joint with Ted Pyne and Salil Vadhan [HPV21])**.** *Let $n \in \mathbb{N}$, let $2^{-n/2} \leq \varepsilon \leq 0.49$, and suppose $G\colon \{0,1\}^s \to \{0,1\}^n$ is an ε-PRG for unbounded-width length-n permutation ROBPs. Then*

$$s \geq \Omega\left(\log\left(\frac{n}{\log(1/\varepsilon)}\right) \cdot \log(1/\varepsilon)\right).$$

When $\varepsilon \geq 2^{-n^{0.99}}$, the lower bound in Theorem 5.1.19 is $\Omega(\log n \cdot \log(1/\varepsilon))$. This is the first known case where a *multiplicative* penalty of $\log(1/\varepsilon)$ is unavoidable. For example, for constant error, the optimal seed length is $\widetilde{O}(\log n)$, but for error $1/n$, the optimal seed length is $\Theta(\log^2 n)$. With respect to fooling low-depth circuits, similar bounds were identified previously for specific families of PRGs, such as k-wise independent generators [LV96] or small-bias generators [DETT10], but Theorem 5.1.19 applies to any PRG whatsoever.

As discussed in Section 1.4, our lower bound helps to clarify the strengths of HSGs

and WPRGs. In the paper [HPV21], we give a probabilistic argument showing that there are $(1/n)$-HSGs for unbounded-width length-n permutation ROBPs with seed length $O(\log n)$, and as mentioned previously, Pyne and Vadhan recently gave an explicit construction of a $(1/n)$-WPRG for such programs with seed length $\widetilde{O}(\log^{3/2} n)$ [PV21]. Thus, when it comes to fooling unbounded-width permutation ROBPs, HSGs and WPRGs are strictly more powerful than unweighted PRGs.

When ε is very small, there is a slight gap between our upper bound (Theorem 5.1.2) and our lower bound (Theorem 5.1.19). It turns out to be possible to slightly improve the *upper* bound [HPV21]. The upshot is that for all $\varepsilon \leq 1/\log n$, we have an explicit ε-PRG for permutation ROBPs with asymptotically optimal seed length. When ε is constant, there is another gap in our bounds ($O(\log n \cdot \log\log n)$ vs. $\Omega(\log n)$), and it remains an interesting open problem to identify the optimal seed length.

Permutation ROBPs for testing permutations

The first step in the proof of Theorem 5.1.19 is to identify a suitable "hard family" of permutation ROBPs.

Lemma 5.1.20. *Let n be an even positive integer, and let $\pi\colon \{0,1\}^{n/2} \to \{0,1\}^{n/2}$ be a permutation. There is a width-$2^{n/2}$ length-n permutation ROBP f such that*

$$f(x,y) = 1 \iff \pi(x) = y.$$

Proof. We identify the state space $[w]$ of f with $\mathbb{F}_2^{n/2}$. Let $e_1, \ldots, e_{n/2}$ be the standard basis vectors of $\mathbb{F}_2^{n/2}$. The transitions of f are defined by

$$\mathrm{next}_i(u,b) = \begin{cases} u + b \cdot e_i & \text{if } i \leq n/2 \\ \pi^{-1}(\pi(u) + b \cdot e_i) & \text{if } i > n/2. \end{cases}$$

The permutation condition is satisfied, because

$$\text{next}_i(\text{next}_i(u, b), b) = u.$$

The start state is the 0 vector, and the accept state is $\pi^{-1}(0)$. Clearly, $f(x, y)$ passes through the state x in layer $n/2$ and arrives at state $\pi^{-1}(\pi(x) + y)$ in layer n. □

From Lemma 5.1.20, it readily follows that a random function with seed length $o(n)$ fails to be a good PRG for permutation ROBPs. The idea is that with high probability, the seed can be recovered from the first half of the generator's output and from the second half of the generator's output, which implies that there is some permutation π such that every output (x, y) of the generator satisfies $\pi(x) = y$. See the paper [HPV21] for details.

Intuitively, the reason the INW generator does better than a random function is that after generating the first half of the output, it uses some fresh randomness to generate the second half of the output, so each half is fairly unpredictable given the other. To prove our seed length lower bound (Theorem 5.1.19), we will show that *every* PRG for permutation ROBPs has a similar property: the second half of the PRG's output must have a considerable amount of entropy given the first half, and vice versa. The reason will be that if each half is somewhat predictable given the other, then we will find a permutation π such that the output (X, Y) of the PRG has a noticeable chance of satisfying $\pi(X) = Y$. The corresponding ROBP from Lemma 5.1.20 therefore has a noticeable acceptance probability under the pseudorandom distribution, whereas it has an extremely low acceptance probability under the uniform distribution, contradicting the PRG correctness condition. Recursing will give a lower bound on the total entropy of the PRG's output, hence also on the seed length of the PRG.

Permutations from matching theory

To find a suitable permutation π, we will use a tool from matching theory. Recall Kőnig's theorem: in any bipartite graph, there exists a matching and a $\{0,1\}$-assignment to the vertices such that every edge in the graph has at least one endpoint marked 1 and the size of the matching is equal to the Hamming weight of the assignment. Egerváry's theorem is a generalization for graphs whose edges have nonnegative integer edge weights. We rely on a variant for graphs whose edges have nonnegative *real* edge weights:

Lemma 5.1.21 (Kőnig-Egerváry theorem for fractional edge weights)**.** *Let $N \in \mathbb{N}$, and let $p\colon [N] \times [N] \to [0,\infty)$. There exist a permutation $\pi\colon [N] \to [N]$ and functions $q, r\colon [N] \to [0,\infty)$ such that*

$$\forall x, y \in [N], p(x,y) \leq q(x) + r(y) \tag{5.7}$$

and

$$\sum_{x\in[N]} p(x,\pi(x)) = \sum_{x\in[N]} q(x) + \sum_{y\in[N]} r(y). \tag{5.8}$$

(To see the connection with matchings, think of p as the weight function on the edges of $K_{N,N}$. The permutation π identifies a perfect matching, and q and r define an assignment to the vertices.) Lemma 5.1.21 is well known; a proof can be found in Schrijver's text [Sch03, Theorem 17.1]. For clarity, we include a proof based on LP duality.

Proof. Consider the problem of finding values $z(x,y)$ to maximize $\sum_{x,y\in[N]} p(x,y)\cdot z(x,y)$,

subject to the constraints

$$\begin{aligned} \forall x, \forall y, z(x,y) &\geq 0 \\ \forall x, \sum_{y \in [N]} z(x,y) &\leq 1 \\ \forall y, \sum_{x \in [N]} z(x,y) &\leq 1. \end{aligned}$$

This is a linear program. We claim that every basic feasible solution is an integer solution, i.e., $z(x,y) \in \{0,1\}$. To prove it, consider some feasible solution z, and think of z as a weight function on the edges of $K_{N,N}$. Let E be the set of edges with fractional weight (i.e., $z(x,y) \in (0,1)$), and let V be the set of vertices such that the sum of the weights incident to that vertex is strictly less than 1.

Suppose $E \neq \varnothing$. For any edge $(x,y) \in E$, either $x \in V$ or else there is another edge $(x,y') \in E$. Similarly, either $y \in V$ or else there is another edge $(x',y) \in E$. Inductively, this implies that there exists a path P consisting entirely of edges in E such that either (a) P is a cycle or else (b) the two endpoints of P are both in V.

Define $\delta(x,y)$ to be 0 for edges not on P and alternately ± 1 for edges on P. This makes sense even if P is a cycle, because the graph is bipartite, so the cycle has even length. Then for all sufficiently small $\varepsilon > 0$, the functions $z'(x,y) = z(x,y) + \varepsilon\delta(x,y)$ and $z''(x,y) = z(x,y) - \varepsilon\delta(x,y)$ are both feasible solutions to the LP. Since z is the average of z' and z'', z must not be a *basic* feasible solution.

When z is $\{0,1\}$-valued, it identifies a subset of the edges, and the constraints say that the subset is a matching. Since p is nonnegative, there is an optimal solution that is a *perfect* matching, corresponding to some permutation $\pi\colon [N] \to [N]$. The value of the objective function is $\sum_{x \in [N]} p(x, \pi(x))$.

The dual LP is the problem of finding values $q(x), r(y)$ to minimize $\sum_{x \in [N]} q(x) +$

$\sum_{y \in [N]} r(y)$, subject to the constraints

$$\forall x, q(x) \geq 0$$
$$\forall y, r(y) \geq 0$$
$$\forall x, \forall y, q(x) + r(y) \geq p(x, y).$$

By strong LP duality, there is a solution with the same value as the primal solution, giving us our desired q and r. □

Permutations for predicting random variables

Now we turn to showing that if each of X and Y is somewhat predictable given the other, then there is a noticeable chance that $\pi(X) = Y$, where π is a permutation that we will obtain from Lemma 5.1.21. We begin by reviewing the standard notion of Shannon entropy.

Definition 5.1.22. *Let X be a discrete random variable. The* Shannon entropy *of X is defined by*

$$\mathrm{H}[X] = \mathop{\mathbb{E}}_{x \sim X} \left[\log \left(\frac{1}{\Pr[X = x]} \right) \right].$$

For two random variables X, Y, the joint entropy *$H[X, Y]$ is the entropy of the pair (X, Y), and the* conditional entropy *is defined by*

$$\mathrm{H}[X \mid Y] = \mathop{\mathbb{E}}_{y \sim Y}[\mathrm{H}[X \mid Y = y]] = \mathop{\mathbb{E}}_{\substack{x \sim X \\ y \sim Y}} \left[\log \left(\frac{1}{\Pr[X = x \mid Y = y]} \right) \right].$$

Lemma 5.1.23. *Let X and Y be random variables distributed over $[N]$. There exists a permutation $\pi \colon [N] \to [N]$ such that*

$$\Pr[\pi(X) = Y] \geq 2^{-\mathrm{H}[X|Y] - \mathrm{H}[Y|X]}.$$

For comparison, if we allow arbitrary functions π (not necessarily permutations), then the maximum value of $\Pr[\pi(X) = Y]$ is closely connected to the conditional *min-entropy* of Y given X [DORS08].

Proof. Let $p(x, y) = \Pr[(X, Y) = (x, y)]$. Applying Lemma 5.1.21, we get a permutation π and two functions $q, r \colon [N] \to [0, \infty)$. Then

$$
\begin{aligned}
&2^{-\mathrm{H}[X|Y]-\mathrm{H}[Y|X]} \\
&\quad = 2^{\mathbb{E}_{(x,y)\sim(X,Y)}[\log(\Pr[X=x|Y=y]\cdot\Pr[Y=y|X=x])]} \\
&\quad \le \mathop{\mathbb{E}}_{(x,y)\sim(X,Y)} [\Pr[X = x \mid Y = y] \cdot \Pr[Y = y \mid X = x]] && \text{(Jensen)} \\
&\quad = \sum_{\substack{x\in\mathrm{Supp}(X)\\ y\in\mathrm{Supp}(Y)}} p(x, y) \cdot \Pr[X = x \mid Y = y] \cdot \Pr[Y = y \mid X = x] \\
&\quad \le \sum_{\substack{x\in\mathrm{Supp}(X)\\ y\in\mathrm{Supp}(Y)}} (q(x) + r(y)) \cdot \Pr[X = x \mid Y = y] \cdot \Pr[Y = y \mid X = x] && \text{(Eq. (5.7))} \\
&\quad \le \sum_{\substack{x\in\mathrm{Supp}(X)\\ y\in\mathrm{Supp}(Y)}} (q(x) \cdot \Pr[Y = y \mid X = x] + r(y) \cdot \Pr[X = x \mid Y = y]) \\
&\quad = \sum_{x\in\mathrm{Supp}(X)} q(x) + \sum_{y\in\mathrm{Supp}(Y)} r(y) \\
&\quad \le \sum_{x\in[N]} q(x) + \sum_{y\in[N]} r(y) \\
&\quad = \sum_{x\in[N]} p(x, \pi(x)) && \text{(Eq. (5.8))} \\
&\quad = \Pr[\pi(X) = Y]. && \square
\end{aligned}
$$

Entropy lower bounds for pseudorandom distributions

Lemma 5.1.23 will imply an entropy lower bound via the standard *chain rule* for Shannon entropy.

Claim 5.1.24 (Chain Rule)**.** *If X and Y are discrete random variables, then*

$$\mathrm{H}[X,Y] = \mathrm{H}[X] + \mathrm{H}[Y \mid X] = \mathrm{H}[Y] + \mathrm{H}[X \mid Y].$$

Now we show that if a distribution over $\{0,1\}^n$ fools unbounded-width permutation ROBPs, then the distribution has $\Omega(\log(1/\varepsilon))$ bits of entropy above and beyond the average entropy of the first $n/2$ bits and the second $n/2$ bits.

Lemma 5.1.25. *Let n be an even positive integer and let X and Y be random variables distributed over $\{0,1\}^{n/2}$. If (X,Y) fools unbounded-width permutation ROBPs with error $\varepsilon \geq 2^{-n/2}$, then*

$$\mathrm{H}[X,Y] \geq \frac{1}{2}\left(\mathrm{H}[X] + \mathrm{H}[Y] + \log\left(\frac{1}{2\varepsilon}\right)\right).$$

Proof. Let π be the permutation guaranteed by Lemma 5.1.23, and let f be the permutation ROBP corresponding to π guaranteed by Lemma 5.1.20. Then

$$2^{-\mathrm{H}[X|Y]-\mathrm{H}[Y|X]} \leq \Pr[\pi(X) = Y] = \mathbb{E}[f(X,Y)] \leq \mathbb{E}[f] + \varepsilon = 2^{-n/2} + \varepsilon \leq 2\varepsilon.$$

Therefore, $\mathrm{H}[X \mid Y] + \mathrm{H}[Y \mid X] \geq \log(\frac{1}{2\varepsilon})$. Finally, by the chain rule,

$$\mathrm{H}[X,Y] = \frac{1}{2}\left(\mathrm{H}[X] + \mathrm{H}[Y] + \mathrm{H}[X \mid Y] + \mathrm{H}[Y \mid X]\right). \qquad \square$$

Corollary 5.1.26. *Let $\varepsilon > 0$, and for $i \geq 0$, let $n_i = \lceil \log(1/\varepsilon) \rceil \cdot 2^i$. Let X be a distribution over $\{0,1\}^{n_i}$ that fools unbounded-width permutation branching programs with error ε. Then $\mathrm{H}[X] \geq \frac{i}{2} \cdot \log\left(\frac{1}{2\varepsilon}\right)$.*

Proof. We proceed by induction on i. In the base case $i = 0$, we are claiming that $\mathrm{H}[X] \geq 0$, which is trivial. For the inductive step, consider a pair (X, Y), where $|X| = |Y| = n_i$, such that (X, Y) fools unbounded-width permutation ROBPs with error ε. Since $\varepsilon \geq 2^{-n_i}$, we may apply Lemma 5.1.25 to get

$$\mathrm{H}[X, Y] \geq \frac{1}{2}\left(\mathrm{H}[X] + \mathrm{H}[Y] + \log\left(\frac{1}{2\varepsilon}\right)\right).$$

Since a permutation ROBP can elect to ignore some of its input bits, X and Y must each individually fool unbounded-width permutation ROBPs with error ε. Therefore, by the induction hypothesis, $\mathrm{H}[X]$ and $\mathrm{H}[Y]$ are both at least $\frac{i}{2} \cdot \log\left(\frac{1}{2\varepsilon}\right)$. Therefore,

$$\mathrm{H}[X, Y] \geq \frac{i+1}{2} \cdot \log\left(\frac{1}{2\varepsilon}\right). \qquad \square$$

Proof of Theorem 5.1.19. Let $i = \lfloor \log(n/\lceil \log(1/\varepsilon) \rceil) \rfloor$, so $n/2 \leq n_i \leq n$. Let $X = G(U_s)_{1\ldots n_i}$. Then X fools unbounded-width length-(n_i) permutation branching programs with error ε, so by Corollary 5.1.26, we have $\mathrm{H}[X] \geq \frac{i}{2} \cdot \log\left(\frac{1}{2\varepsilon}\right)$. Applying a deterministic function can only decrease entropy, so

$$s = \mathrm{H}[U_s] \geq \mathrm{H}[X] \geq \frac{1}{2}\left\lfloor \log\left(\frac{n}{\lceil \log(1/\varepsilon) \rceil}\right)\right\rfloor \cdot \log\left(\frac{1}{2\varepsilon}\right) \geq \Omega\left(\log\left(\frac{n}{\log(1/\varepsilon)}\right) \cdot \log(1/\varepsilon)\right),$$

where the last step uses our assumption $2^{-n/2} \leq \varepsilon \leq 0.49$. $\square$

5.2 PRGs for Read-Once $\mathbf{AC}^0$

In this section, we present a near-optimal PRG for *read-once* $\mathbf{AC}^0$ *formulas*, i.e., read-once formulas with unbounded-fan-in AND/OR gates with negations allowed at the inputs. We do not assume that the variables are ordered in any way. This PRG is joint work with Dean

Doron and Pooya Hatami [DHH19].

Theorem 5.2.1 (Joint with Dean Doron and Pooya Hatami [DHH19])**.** *For every $n, d \in \mathbb{N}$ and $\varepsilon > 0$, there is an explicit ε-PRG for depth-d read-once $\mathbf{AC}^0$ formulas with seed length*

$$\log(n/\varepsilon) \cdot O(d \log\log(n/\varepsilon))^{2d+2}.$$

When the depth d is a constant, our seed length is $\widetilde{O}(\log(n/\varepsilon))$. The optimal seed length would be $O(\log(n/\varepsilon))$, with no dependence on d.

Motivation

Read-once $\mathbf{AC}^0$ is connected to $\mathbf{L}$ vs. $\mathbf{BPL}$ via the concept of an *arbitrary-order ROBP* [BPW11; SVW17; CHRT18; FK18].

Definition 5.2.2. *A width-w length-n* arbitrary-order ROBP *is a function $f\colon \{0,1\}^n \to \{0,1\}$ of the form*

$$f(x) = g(x_{\pi(1)}, \ldots, x_{\pi(n)}),$$

where g is a width-w length-n ROBP and π is a permutation on $[n]$.

The arbitrary-order ROBP model is arguably more natural than the standard ROBP model, especially to those coming from a background in circuit complexity. Nisan's PRG [Nis92] provably fails to fool arbitrary-order ROBPs [Tzu09], and one reason to study PRGs for arbitrary-order ROBPs is to force ourselves to do something completely different than Nisan's PRG, with the hope that the insights gained along the way will yield benefits even for the standard ordered model. Building on several prior works [RSV13; HLV18; CHRT18], Forbes and Kelley designed PRGs for arbitrary-order ROBPs with seed length $O(\log^3 n)$ in the polynomial-width case and $\widetilde{O}(\log^2 n)$ in the constant-width case [FK18]. Their PRG

is based on *pseudorandom restrictions*, which is indeed quite different than Nisan's PRG [Nis92] and other classic PRGs for space-bounded computation [INW94; NZ96].

Our PRG for read-once $\mathbf{AC}^0$ will use pseudorandom restrictions and build on Forbes and Kelley's work [FK18]. This is possible because read-once $\mathbf{AC}^0$ formulas are a special case of constant-width arbitrary-order ROBPs:

Claim 5.2.3 ([CSV15])**.** *If f can be computed by a depth-d read-once $\mathbf{AC}^0$ formula where $d \geq 1$, then f can be computed by a width-$(d+1)$ arbitrary-order ROBP.*

Proof. We proceed by induction on d. Regarding the base case $d = 1$, it is easy to design width-2 ROBPs for any conjunction or disjunction of literals. Now let us consider the inductive step $d > 1$. By negating f if necessary, we may assume without loss of generality that $f(x) = \mathsf{AND}(g_1(x), \ldots, g_m(x))$, where each g_i is a depth-$(d-1)$ read-once $\mathbf{AC}^0$ formula and the g_i formulas read disjoint sets of inputs. By induction, each g_i can be computed by a width-$(d-1)$ arbitrary-order ROBP on the variables that g_i reads. The ROBP for f simulates these ROBPs in sequence. If the ROBP for g_i accepts, then we move to the start state of the ROBP for g_{i+1}. On the other hand, if any ROBP rejects, then we move to a new special reject state, both of whose outgoing edges lead to the reject state in the next layer. □

Looking more carefully at the proof of Claim 5.2.3, one can show [DMRTV21] that read-once $\mathbf{AC}^0$ can be simulated by a special class of constant-width ROBPs called *monotone* ROBPs.

Definition 5.2.4. *An ROBP is* monotone *if each layer can be ordered in such a way that for each $i \in [n]$ and each $b \in \{0,1\}$, the transition function $\mathrm{next}_i(\cdot, b)$ is a monotone function $[w] \to [w]$.*

Monotone ROBPs, which were introduced by Meka and Zuckerman [MZ13], are in some sense "the opposite" of permutation ROBPs [DMRTV21], so this section can be considered complementary to the previous section. Hopefully, studying the two extreme cases of permutation ROBPs and monotone ROBPs will help us to eventually develop better PRGs for unrestricted ROBPs.

Related work

For simplicity, in this section we will mainly focus on the case that the error ε is a constant. As a reminder, for any constant $d \in \mathbb{N}$, our PRG fools depth-d read-once $\mathbf{AC}^0$ formulas with seed length $\widetilde{O}(\log n)$. Prior to our work, Chen, Steinke, and Vadhan gave a PRG for this class with seed length $\widetilde{O}(\log^{d+1} n)$ [CSV15], and then Forbes and Kelley gave a PRG for the more general class of width-$(d+1)$ arbitrary-order ROBPs with seed length $\widetilde{O}(\log^2 n)$ [FK18]. PRGs with near-logarithmic seed length were known only for the depth-2 case [CRS00; DETT10; GMRTV12]; in a follow-up paper to our work, Doron, Hatami, and the author gave a PRG for the depth-2 case (even with parity gates) with the improved seed length $O(\log n) + \widetilde{O}(\log(1/\varepsilon))$ [DHH20].

For context, after a long line of research [AW89; Nis91; Bra10; TX13; Tal17; HS19; ST19a; Kel20], we now have explicit PRGs for *general* polynomial-size depth-d $\mathbf{AC}^0$ circuits with seed length $\widetilde{O}(\log^{d+1} n)$. Meanwhile, Gavinsky, Lovett, and Srinivasan gave a PRG for read-once $\mathbf{ACC}^0$ with seed length $\operatorname{polylog} n$ [GLS12]. Recently, Doron, Meka, Reingold, Tal, and Vadhan gave a PRG for width-$(d+1)$ arbitrary-order monotone ROBPs with seed length $\widetilde{O}(\log n)$ (see Table 5.2), and they showed that this class is *strictly* more powerful than depth-d read-once $\mathbf{AC}^0$ [DMRTV21]. Furthermore, their PRG has a better dependence on d: for super-constant d, our PRG has seed length

$$\log n \cdot O(d \log\log n)^{2d+2},$$

Seed length	Model fooled	Reference
$O(n^{0.001})$	$\mathbf{AC}^0$	[AW89]
$O(\log^{2d+6} n)$	$\mathbf{AC}^0$	[Nis91]
$\widetilde{O}(\log^{d+4} n)$	$\mathbf{AC}^0$	[TX13]
$\widetilde{O}(\log^{d+1} n)$	Read-once $\mathbf{AC}^0$	[CSV15]
$\widetilde{O}(\log^{d+2} n)$	$\mathbf{AC}^0$	[Tal17]
$\widetilde{O}(\log^2 n)$	Arbitrary-order ROBPs	[FK18]
$\widetilde{O}(\log n)$	Read-once $\mathbf{AC}^0$	[DHH19] (Theorem 5.2.1)
$\widetilde{O}(\log^{d+1} n)$	$\mathbf{AC}^0$	[Kel20]
$\widetilde{O}(\log n)$	Arbitrary-order monotone ROBPs	[DMRTV21]

Table 5.2: Explicit PRGs for depth-d read-once $\mathbf{AC}^0$ and relevant more powerful models, where d is a constant.

whereas Doron et al. [DMRTV21] achieve seed length

$$O(\log n \cdot d^2 \cdot (\log \log n)^2).$$

There has also been work on unbounded-depth read-once formulas [BPW11; IMZ19; FK18]. The best result for that model is currently the Forbes-Kelley PRG [FK18], which fools unbounded-depth read-once formulas with constant fan-in over an arbitrary basis with seed length $O(\log^3 n)$. Extending our work to the case of read-k $\mathbf{AC}^0$ is an interesting open problem. For read-k CNFs, explicit PRGs are known with seed length $\log n \cdot \mathrm{poly}(k)$ [KLW10; ST19b].

5.2.1 Random and Pseudorandom Restrictions

As mentioned previously, our PRG for read-once $\mathbf{AC}^0$ is based on pseudorandom restrictions. A *restriction* is a string $x \in \{0,1,\star\}^n$. When $x_i = \star$, the interpretation is that we have not yet decided whether to set $x_i = 0$ or $x_i = 1$. We define the *composition* operator on restrictions by

$$(x \circ x')_i = \begin{cases} x_i & \text{if } x_i \neq \star \\ x'_i & \text{if } x_i = \star. \end{cases}$$

In words, we consult x' as a backup whenever x does not provide a value. If f is a function on $\{0,1\}^n$ and $x \in \{0,1,\star\}^n$, then the *restricted function* $f|_x$ is another function on $\{0,1\}^n$ given by $f|_x(x') = f(x \circ x')$.

A huge body of work in the theory of computing uses the concept of a *random restriction* and in particular the distribution $\mathcal{R}_p$ defined below.

Definition 5.2.5. *Let $n \in \mathbb{N}$ and $p \in [0,1]$. We define $\mathcal{R}_p$ to be the distribution over $\{0,1,\star\}^n$ where the coordinates are independent and each is $\star$ with probability p and a uniform random bit with probability $1-p$.*

Just like a PRG samples a distribution that is essentially indistinguishable from U_n, we will be interested in distributions that are essentially indistinguishable from $\mathcal{R}_p$. We will obtain these pseudorandom restrictions by pseudorandomly generating bits that encode a restriction. To encode a restriction using bitstrings, we use the Res function defined below.

Definition 5.2.6. *We define* $\mathsf{Res}\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1,\star\}^n$ *by*

$$(\mathsf{Res}(y,z))_i = \begin{cases} \star & \text{if } z_i = 1, \\ y_i & \text{if } z_i = 0. \end{cases}$$

Thus z indicates the $\star$ positions and y assigns values to the other positions.

Note that $\mathsf{Res}(U_{2n}) = \mathcal{R}_{1/2}$. To mimic $\mathcal{R}_p$ with $p \ll 1/2$, we will use composition. If R is a distribution over $\{0,1,\star\}^n$ and $t \in \mathbb{N}$, we define $R^{\circ t}$ to be the distribution obtained from drawing t independent samples from R and composing them.

Most of our effort will go toward proving that under a suitable pseudorandom restriction, read-once $\mathbf{AC}^0$ formulas *simplify* in some sense. Using the work of Forbes and Kelley [FK18], we will also ensure that these pseudorandom restrictions approximately preserve the expectation of the formula. Finally, using the work of Meka, Reingold, and Tal [MRT19], we will be able to fool the "simpler" formulas with a short seed. Putting these pieces together will give us our PRG for read-once $\mathbf{AC}^0$. When fooling depth-$(d+1)$ read-once $\mathbf{AC}^0$ formulas, our pseudorandom restriction will actually rely on a PRG for depth-d read-once $\mathbf{AC}^0$ formulas, which we will obtain through recursion. (Gavinsky, Lovett, and Srinivasan used a similar approach for fooling read-once $\mathbf{ACC}^0$ [GLS12].)

5.2.2 Simplification of Read-Once $\mathbf{AC}^0$ under Restrictions

The effect of truly random restrictions

We would like to prove that read-once $\mathbf{AC}^0$ simplifies under pseudorandom restrictions. We begin by reviewing Chen, Steinke, and Vadhan's analysis of the effect of a *truly* random restriction on read-once $\mathbf{AC}^0$ [CSV15]. The following theorem says that any read-once $\mathbf{AC}^0$ formula that is close to a constant is likely to collapse to a constant under a random restriction.

Theorem 5.2.7 ([CSV15])**.** *Let f be a depth-d read-once $\mathbf{AC}^0$ formula that is α-close to a constant. For any $\varepsilon > 0$ and any $p \le 1/(9\log(4^d n/\varepsilon))^d$,*

$$\Pr[f|_{\mathcal{R}_p} \text{ is nonconstant}] \le 2\alpha \cdot p \cdot (9\log(4^d n/\varepsilon))^d + 2\varepsilon.$$

Proof outline. For $q \in [-1,1]$, let $\mathcal{D}_q$ be the product distribution over $\{0,1\}^n$ where each bit has expectation $1/2 + q/2$. First, suppose f is monotone. Let $F = f|_{\mathcal{R}_p}$. Then

$$\begin{aligned}\Pr[F \text{ is nonconstant}] = \mathbb{E}[F(1^n) - F(0^n)] &= \mathbb{E}[f(\mathcal{D}_p)] - \mathbb{E}[f(\mathcal{D}_{-p})] \\ &= \sum_{S \subseteq [n]} \widehat{f}(S) \cdot (\mathbb{E}[\chi_S(\mathcal{D}_p)] - \mathbb{E}[\chi_S(\mathcal{D}_{-p})]) \\ &= \sum_{S \subseteq [n]} \widehat{f}(S) \cdot ((-p)^{|S|} - p^{|S|}) \\ &\leq 2 \sum_{\substack{S \subseteq [n] \\ |S| \text{ odd}}} |\widehat{f}(S)| \cdot p^{|S|}.\end{aligned}$$

Chen, Steinke, and Vadhan showed [CSV15, Theorem 3.1] that

$$\sum_{\substack{S \subseteq [n] \\ S \neq \varnothing}} |\widehat{f}(S)| \cdot p^{|S|} \leq \alpha \cdot p \cdot (9 \log(4^d n/\varepsilon))^d + \varepsilon,$$

completing the proof in this case. Now consider a general f, not necessarily monotone. Since f is read-once, there is some y such that the function $f^{+y}(x) \stackrel{\text{def}}{=} f(x+y)$ is a monotone depth-d read-once $\mathbf{AC}^0$ formula. Clearly, the probability that f becomes constant under $\mathcal{R}_p$ is precisely the same as the probability that f^{+y} becomes constant under $\mathcal{R}_p$. □

NAND formulas

It will be convenient to move to a different representation of read-once $\mathbf{AC}^0$ formulas. A *read-once* NAND *formula* is a tree of NAND gates with literals at the bottom and no repeated variables. This model is equivalent to read-once $\mathbf{AC}^0$ by the following easy lemma.

Lemma 5.2.8. *Let $f\colon \{0,1\}^n \to \{0,1\}$ and $d \in \mathbb{N}$. The following are equivalent.*

1. *f can be computed by a depth-d read-once $\mathbf{AC}^0$ formula.*

2. *Either f or $\neg f$ can be computed by a depth-d read-once* NAND *formula.*

Proof. Starting from a read-once $\mathbf{AC}^0$ formula, we can use De Morgan's laws to replace every OR with NOT $\circ$ AND $\circ$ NOT. This gives a formula with alternating NOT and AND gates. Introducing a NOT gate at the output if necessary, we can merge NOT $\circ$ AND $=$ NAND to get a depth-d read-once NAND formula.

Conversely, suppose f can be computed by a depth-d read-once NAND formula. By using De Morgan's laws, we can push all the negations to the inputs, giving a depth-d read-once $\mathbf{AC}^0$ formula. This, in turn, implies that $\neg f$ can be computed by a depth-d read-once $\mathbf{AC}^0$ formula as well. □

In the context of NAND formulas, it makes sense to focus on the event of collapsing to the constant 1 rather than the constant 0.

Corollary 5.2.9. *Let f be a depth-d read-once* NAND *formula with $\mathbb{E}[f] \geq 1 - \alpha$. For any $\varepsilon > 0$ and any $p \leq 0.5/(9\log(4^d n/\varepsilon))^d$,*

$$\Pr_{R \leftarrow \mathcal{R}_p}[f|_R \not\equiv 1] \leq 2\alpha + 2\varepsilon.$$

Proof. For this value of p, Theorem 5.2.7 implies that

$$\Pr[f|_R \text{ is nonconstant}] \leq \alpha + 2\varepsilon.$$

Furthermore, clearly, $\Pr[f|_R \equiv 0] \leq \alpha$. The claim follows by the union bound. □

Derandomizing simplification

We would like to derandomize Corollary 5.2.9, i.e., we would like to show that a similar conclusion holds under a *pseudorandom* restriction. The Fourier analysis in the proof of

Theorem 5.2.7 does not seem particularly amenable to derandomization. Instead, we will show that the condition in the conclusion of Corollary 5.2.9 can itself be tested in read-once $\mathbf{AC}^0$, hence Corollary 5.2.9 can be derandomized using any PRG for read-once $\mathbf{AC}^0$. Later, we will use this fact recursively to actually construct a PRG for read-once $\mathbf{AC}^0$.

The next claim shows that the condition that a formula collapses to the constant 1 under some restriction can be tested by another read-once $\mathbf{AC}^0$ formula that reads the bits encoding the restriction. For the first reading of Claim 5.2.10, it might be easiest to ignore u (i.e., assume $u = \star^n$).

Claim 5.2.10. *Let f be a depth-d read-once* NAND *formula on the variables $x_1, \ldots, x_n$, and let $u \in \{0,1,\star\}^n$. There is some depth-$(d+1)$ read-once $\mathbf{AC}^0$ formula g on the variables $y_1, \ldots, y_n, z_1, \ldots, z_n$ such that*

$$g(y,z) = 1 \iff f|_{\mathsf{Res}(y,z)\circ u} \equiv 1.$$

Proof. We will prove more generally that for each $b \in \{0,1\}$, there is some depth-$(d+1)$ read-once $\mathbf{AC}^0$ formula g on the variables $y_1, \ldots, y_n, z_1, \ldots, z_n$ such that

$$g(y,z) = 1 \iff f|_{\mathsf{Res}(y,z)\circ u} \equiv b.$$

In the base case $d = 0$, we are looking at a single literal, $f(x) = x_i$ or $f(x) = \neg x_i$. Applying the definition of Res, we have

$$\begin{aligned}
(x_i)|_{\mathsf{Res}(y,z)\circ u} \equiv b &\iff z_i = 1 \vee y_i = b && \text{if } u_i = b \\
(x_i)|_{\mathsf{Res}(y,z)\circ u} \equiv b &\iff z_i = 0 \wedge y_i = b && \text{if } u_i \neq b.
\end{aligned}$$

Either way, this is a depth-1 read-once $\mathbf{AC}^0$ formula in the variables y_i, z_i. As for the other

case $f(x) = \neg x_i$, under any restriction r, clearly

$$(\neg x_i)|_r \equiv b \iff (x_i)|_r \equiv 1 - b,$$

so once again we get a depth-1 read-once $\mathbf{AC}^0$ formula.

Now for the inductive step $d \geq 1$, we have $f(x) = \mathsf{NAND}(h_1(x), \ldots, h_m(x))$. Observe that

$$\mathsf{NAND}(h_1, \ldots, h_m) \equiv 0 \iff \bigwedge_{i=1}^{m} (h_i \equiv 1)$$
$$\mathsf{NAND}(h_1, \ldots, h_m) \equiv 1 \iff \bigvee_{i=1}^{m} (h_i \equiv 0).$$

By induction, the conditions $h_i|_{\mathsf{Res}(y,z) \circ u} \equiv 1$ and $h_i|_{\mathsf{Res}(y,z) \circ u} \equiv 0$ can each be computed by a depth-d read-once $\mathbf{AC}^0$ formula, so in each case, the right-hand side is indeed a depth-$(d+1)$ read-once $\mathbf{AC}^0$ formula. □

Now we are ready to derandomize Corollary 5.2.9, i.e., we will show that if we start with a read-once NAND formula that is biased toward 1 and we hit it with $O(\log \log n)$ pseudorandom restrictions (each with $\star$-probability $1/2$), then with high probability the formula will collapse to the constant 1 function.

Corollary 5.2.11 (Derandomization of Corollary 5.2.9)**.** *Let $d \leq \log n$. Let f be a depth-d read-once* NAND *formula with $\mathbb{E}[f] \geq 1 - \alpha$. Let (Y, Z) be a distribution over $\{0,1\}^{2n}$ that γ-fools depth-$(d+1)$ read-once $\mathbf{AC}^0$ formulas. For any $\varepsilon > 0$, there is a value $t = O(d \log \log(n/\varepsilon))$ such that*

$$\Pr_{R \leftarrow \mathsf{Res}(Y,Z)^{\circ t}} [f|_R \not\equiv 1] \leq 2\alpha + 2\varepsilon + t\gamma.$$

Proof. Let us show by induction on t that

$$\Pr_{R\leftarrow \mathsf{Res}(Y,Z)^{\circ t}}[f|_R \not\equiv 1] \leq \Pr_{R\leftarrow \mathcal{R}_{2^{-t}}}[f|_R \not\equiv 1] + t\gamma.$$

In the base case $t = 0$, this is trivial, because on both sides of the inequality, $R = \star^n$. For the inductive step $t > 0$, letting U and U' denote independent uniform n-bit strings, we have

$$\begin{aligned}
\Pr_{R\leftarrow \mathsf{Res}(Y,Z)^{\circ t}}[f|_R \not\equiv 1] &= \mathop{\mathbb{E}}_{R'\leftarrow \mathsf{Res}(Y,Z)^{\circ (t-1)}}\left[\Pr_{R''\leftarrow \mathsf{Res}(Y,Z)}[f|_{R''\circ R'} \not\equiv 1]\right] \\
&\leq \mathop{\mathbb{E}}_{R'\leftarrow \mathsf{Res}(Y,Z)^{\circ (t-1)}}\left[\Pr_{R''\leftarrow \mathsf{Res}(U,U')}[f|_{R''\circ R'} \not\equiv 1]\right] + \gamma \qquad \text{(Claim 5.2.10)} \\
&= \mathop{\mathbb{E}}_{R''\leftarrow \mathsf{Res}(U,U')}\left[\Pr_{R'\leftarrow \mathsf{Res}(Y,Z)^{\circ (t-1)}}[f|_{R''\circ R'} \not\equiv 1]\right] + \gamma \\
&\leq \mathop{\mathbb{E}}_{R''\leftarrow \mathsf{Res}(U,U')}\left[\Pr_{R'\leftarrow \mathsf{Res}(U,U')^{\circ (t-1)}}[f|_{R''\circ R'} \not\equiv 1]\right] + t\gamma \qquad \text{(Induction)} \\
&= \Pr_{R\leftarrow \mathsf{Res}(U,U')^{\circ t}}[f|_R \not\equiv 1] + t\gamma \\
&= \Pr_{R\leftarrow \mathcal{R}_{2^{-t}}}[f|_R \not\equiv 1] + t\gamma.
\end{aligned}$$

Finally, there is a value $t = O(d \log\log(n/\varepsilon))$ such that $2^{-t} \leq 0.5/(9\log(4^d n/\varepsilon))^d$, so by Corollary 5.2.9, we get

$$\Pr_{R\leftarrow \mathcal{R}_{2^{-t}}}[f|_R \not\equiv 1] \leq 2\alpha + 2\varepsilon. \qquad \square$$

The Δ complexity measure

Corollary 5.2.11 has two serious weaknesses. First, it assumes that $\mathbb{E}[f] \approx 1$. Second, to make a formula simplify, it requires us to already have a PRG for *deeper* formulas. We address both of these weaknesses by studying a more subtle notion of "simplification" defined below.

Definition 5.2.12. *For a read-once* NAND *formula f, we define $\Delta(f)$ to be the maximum fan-in of any gate other than the root.*

We now show that with respect to the Δ complexity measure, formulas are likely to simplify under pseudorandom restrictions generated using a PRG for *shallower* formulas. This lemma does not require that the formula is close to a constant, although it does require that the formula is *internally unpredictable*, a notion that we define next. If f is a read-once NAND formula, a *subformula* of f is another read-once NAND formula obtained by taking some gate in f together with all of its descendants.

Definition 5.2.13. *Let f be a read-once NAND formula and let $\gamma > 0$. We say that f is* γ-internally-unpredictable *if every nonconstant subformula g of f satisfies $\mathbb{E}[g] \in [\gamma, 1-\gamma]$.*

Lemma 5.2.14. *Let $1 \le d \le \log n$, let f be a depth-$(d+1)$ read-once NAND formula, and let $\delta > 0$. For a suitable $\gamma = (\delta/n)^{O(1)}$, let (Y, Z) be a distribution over $\{0,1\}^{2n}$ that fools depth-d read-once $\mathbf{AC}^0$ formulas with error γ, and assume f is γ-internally-unpredictable. Then for a suitable $t = O(d \log\log(n/\delta))$, we have*

$$\Pr_{R \leftarrow \mathsf{Res}(Y,Z)^{\circ t}} \left[\Delta(f|_R) \le O\left(\sqrt{\Delta(f)} \cdot \log^2(n/\delta) \right) \right] \ge 1 - \delta.$$

The proof technique is similar to the analysis of Gopalan, Meka, Reingold, Trevisan, and Vadhan [GMRTV12] regarding read-once CNFs, as well as Chen, Steinke, and Vadhan's analysis of read-once $\mathbf{AC}^0$ under a truly random restriction [CSV15].

Proof. Let g be a proper subformula of f, say $g(x) = \mathsf{NAND}(h_1(x), \dots, h_m(x))$. We shall bucket the children of g according to their expectations. Let $\mathcal{H}_i$ be the set of children with expectation roughly $1 - 2^{-i}$; more precisely,

$$\mathcal{H}_i = \{h_j : 1 - 2^{-i} \le \mathbb{E}[h_j] < 1 - 2^{-(i+1)}\}.$$

Since f is γ-internally-unpredictable, we have $\mathcal{H}_0 \cup \cdots \cup \mathcal{H}_{\log(1/\gamma)} = \{h_1, \dots, h_m\}$. Let us focus on a single bucket $\mathcal{H} = \mathcal{H}_i$. Let $\alpha = 2^{-i}$, so each $h_j \in \mathcal{F}$ satisfies $1 - \alpha \le \mathbb{E}[h_j] \le 1 - \alpha/2$.

Define a random variable

$$L = |\{h_j \in \mathcal{H} : h_j|_R \not\equiv 1\}|.$$

Then for values M and k to be chosen later, we have

$$\begin{aligned}
\Pr[L \geq M] &= \Pr\left[\binom{L}{k} \geq \binom{M}{k}\right] && \text{(Pascal's rule)} \\
&\leq \frac{\mathbb{E}[\binom{L}{k}]}{\binom{M}{k}} && \text{(Markov's inequality)} \\
&= \frac{1}{\binom{M}{k}} \cdot \sum_{S \subseteq \mathcal{H}, |S|=k} \Pr[\forall h_j \in S, h_j|_R \not\equiv 1].
\end{aligned}$$

Now, fix some set $S \subseteq \mathcal{F}$ with $|S| = k$. Define $h(x) = \mathsf{OR}(S)$. Observe that

$$h|_R \equiv 1 \iff \exists h_j \in S, h_j|_R \equiv 1.$$

Furthermore, $\mathbb{E}[h] \geq 1 - \alpha^k$. Finally, h can be computed by a read-once depth-$(d-1)$ NAND formula (the same depth as each individual h_j), because

$$\begin{aligned}
\mathsf{OR} \circ \mathsf{NAND} &= (\mathsf{NOT} \circ \mathsf{AND} \circ \mathsf{NOT}) \circ (\mathsf{NOT} \circ \mathsf{AND}) \\
&= \mathsf{NOT} \circ \mathsf{AND} \circ \mathsf{AND} \\
&= \mathsf{NOT} \circ \mathsf{AND} \\
&= \mathsf{NAND}.
\end{aligned}$$

Therefore, by Corollary 5.2.11,

$$\Pr[h|_R \not\equiv 1] \leq 2\alpha^k + (t+2)\gamma < 2\alpha^k + 2t\gamma.$$

Thus,

$$\begin{aligned}\Pr[L \geq M] \leq \frac{1}{\binom{M}{k}} \cdot \sum_{S \subseteq \mathcal{H}, |S|=k} (2\alpha^k + 2t\gamma) &\leq \frac{\binom{|\mathcal{H}|}{k}}{\binom{M}{k}} \cdot (2\alpha^k + 2t\gamma) \\ &\leq \left(\frac{e \cdot |\mathcal{H}|}{M}\right)^k \cdot (2\alpha^k + 2t\gamma) \\ &= 2\left(\frac{e \cdot |\mathcal{H}| \cdot \alpha}{M}\right)^k + 2t\gamma \cdot \left(\frac{e \cdot |\mathcal{H}|}{M}\right)^k.\end{aligned}$$

To bound the first term, note that

$$\gamma \leq \mathbb{E}[\neg g] = \prod_{j=1}^{m} \mathbb{E}[h_i] < (1 - \alpha/2)^{|\mathcal{H}|} \leq e^{-\alpha/2 \cdot |\mathcal{H}|},$$

and hence

$$\begin{aligned}&|\mathcal{H}| < (2/\alpha) \cdot \log(1/\gamma) \\ \implies &\left(\frac{e \cdot |\mathcal{H}| \cdot \alpha}{M}\right)^k \leq \left(\frac{2e \cdot \log(1/\gamma)}{M}\right)^k.\end{aligned}$$

To bound the second term, note that trivially $|\mathcal{H}| \leq \Delta(f)$. Therefore, setting $M = 2e \cdot \sqrt{\Delta(f)} \log(1/\gamma)$, we get an overall bound of

$$\Pr[L \geq M] \leq \frac{2}{\Delta(f)^{k/2}} + 2t\gamma \cdot \Delta(f)^{k/2}.$$

For a suitable $k = \Theta(\log(n/\delta)/\log(\Delta(f)))$, the bound comes out to $\frac{\delta}{2dn^2} + 8tdn^2\gamma/\delta$, which is at most δ/n^2 for a suitable choice of $\gamma = \Theta(\frac{\delta^2}{tdn^2}) = (\delta/n)^{O(1)}$.

There are at most n nonempty buckets $\mathcal{H}$, and there are at most nd proper subformulas

g. Therefore, by a union bound, with probability at least $1 - \delta$, we have

$$\Delta(f|_R) \leq M \cdot (1 + \log(1/\gamma)) = O\left(\sqrt{\Delta(f)} \cdot \log^2(1/\gamma)\right) = O\left(\sqrt{\Delta(f)} \cdot \log^2(n/\delta)\right). \quad \square$$

Sandwiching by internally unpredictable formulas

We would like to remove the assumption that f is internally unpredictable in Lemma 5.2.14. Intuitively, we ought to be able to assume without loss of generality that our formula is internally unpredictable, because a subformula that is close to a constant hardly affects the computation. This intuition can be justified using the following lemma by Chen, Steinke, and Vadhan [CSV15]. If f_ℓ, f, and f_u are all functions mapping $\{0,1\}^n \to \mathbb{R}$, we say that f_ℓ and f_u *ε-sandwich* f if $f_\ell \leq f \leq f_u$ and $\mathbb{E}[f_u] \leq \mathbb{E}[f_\ell] + \varepsilon$.

Lemma 5.2.15 ([CSV15; DHH19])**.** *Let f be a read-once depth-d NAND formula, let $\varepsilon > 0$, and let $\gamma = \frac{\varepsilon^2}{4dn^2}$. There exist γ-internally-unpredictable read-once NAND formulas f_ℓ, f_u that ε-sandwich f such that the underlying tree structures of f_ℓ and f_u are subtrees of the underlying tree structure of f.*

We reproduce the proof for clarity, since Chen, Steinke, and Vadhan did not discuss the exact statement of Lemma 5.2.15.

Proof. Let $\mathrm{size}(f)$ denote the number of NAND gates in f. We will achieve sandwiching error $n \cdot \sqrt{\gamma} + \mathrm{size}(f) \cdot \gamma < \varepsilon$. We proceed by induction on $\mathrm{size}(f)$. In the base case $\mathrm{size}(f) = 0$, f is just a constant or a literal, which is trivially γ-internally-unpredictable, so we can take $f_\ell = f_u = f$.

Now consider the case $\mathrm{size}(f) > 0$, say $f(x) = \mathsf{NAND}(g_1(x), \ldots, g_m(x))$. By induction,

for each i, we get sandwiching formulas g_i^ℓ and g_i^u for g_i. We define

$$f_u(x) = \begin{cases} \mathsf{NAND}(g_1^\ell(x), \ldots, g_m^\ell(x)) & \text{if } \mathbb{E}[\mathsf{NAND}(g_1^\ell, \ldots, g_m^\ell)] \leq 1 - \gamma \\ 1 & \text{otherwise,} \end{cases}$$

and we define

$$f_\ell(x) = \mathsf{NAND}((g_i^u(x))_{i \in S}), \text{ where } S \text{ maximizes } \mathbb{E}[f_\ell] \text{ under the constraint } \mathbb{E}[f_\ell] \leq 1 - \gamma.$$

The definition of f_ℓ makes sense, because if nothing else, we can take $S = \{1\}$, in which case $\mathbb{E}[f_\ell] = 1 - \mathbb{E}[g_1^u] \leq 1 - \gamma$ by our induction assumption.

To prove correctness, first note that

$$f_\ell \leq \mathsf{NAND}(g_1^u, \ldots, g_m^u) \leq f \leq \mathsf{NAND}(g_1^\ell, \ldots, g_m^\ell) \leq f_u,$$

and clearly the underlying tree structures of f_ℓ and f_u are subtrees of the underlying tree structure of f. Furthermore, by induction, clearly every nonconstant *proper* subformula g of each sandwiching formula satisfies $\mathbb{E}[g] \in [\gamma, 1 - \gamma]$. By construction, $\mathbb{E}[f_\ell] \leq 1 - \gamma$, and either f_u is constant or else $\mathbb{E}[f_u] \leq 1 - \gamma$. Furthermore, $\mathbb{E}[f_u] \geq 1 - \mathbb{E}[g_1^\ell] \geq \gamma$ by induction, and $\mathbb{E}[f_\ell] \geq 1 - \mathbb{E}[g_1^u] \geq \gamma$ by induction. Thus, f_ℓ and f_u are γ-internally-unpredictable. All that remains is to bound the sandwiching error $\mathbb{E}[f_u - f_\ell]$.

Let n_i be the number of variables feeding into g_i. For the first case, suppose

$$\mathbb{E}[\mathsf{NAND}(g_1^u, \ldots, g_m^u)] \leq 1 - \gamma.$$

Then $f_\ell = \mathsf{NAND}(g_1^u, \ldots, g_m^u)$, so

$$\begin{aligned}
\mathbb{E}[f_u - f_\ell] &\le \mathbb{E}[f_u - \mathsf{NAND}(g_1^\ell, \ldots, g_m^\ell)] + \mathbb{E}[\mathsf{NAND}(g_1^\ell, \ldots, g_m^\ell) - f_\ell] \\
&\le \gamma + \sum_{i=1}^{m} \mathbb{E}[g_i^u - g_i^\ell] && \text{(Union bound)} \\
&\le \gamma + \sum_{i=1}^{m} (n_i \cdot \sqrt{\gamma} + \mathrm{size}(g_i) \cdot \gamma) && \text{(Induction)} \\
&\le n\sqrt{\gamma} + \mathrm{size}(f) \cdot \gamma,
\end{aligned}$$

completing the proof in this case.

For the second case, suppose $\mathbb{E}[\mathsf{NAND}(g_1^u, \ldots, g_m^u)] > 1 - \gamma$. Then $f_u = 1$, so

$$\mathbb{E}[f_u - f_\ell] = 1 - \mathbb{E}[f_\ell] = \prod_{i \in S} \mathbb{E}[g_i^u],$$

where S was chosen to minimize $\prod_{i \in S} \mathbb{E}[g_i^u]$ under the constraint $\prod_{i \in S} \mathbb{E}[g_i^u] \ge \gamma$. If there is some i such that $\mathbb{E}[g_i^u] \le \sqrt{\gamma}$, then one valid choice is $S = \{i\}$, showing that indeed, $\mathbb{E}[f_u - f_\ell] \le \sqrt{\gamma}$, completing the proof in this case.

Assume now that for every i, we have $\mathbb{E}[g_i^u] > \sqrt{\gamma}$. On the other hand, $\prod_{i=1}^{m} \mathbb{E}[g_i^u] = 1 - \mathbb{E}[\mathsf{NAND}(g_1^u, \ldots, g_m^u)] < \gamma$. Therefore, there must be some $j \in [m]$ such that $\prod_{i=1}^{j} \mathbb{E}[g_i^u] \in [\gamma, \sqrt{\gamma}]$. One valid choice is $S = [j]$, showing once again that $\mathbb{E}[f_u - f_\ell] \le \sqrt{\gamma}$. □

Simplification of general read-once $\mathbf{AC^0}$ formulas

At long last, the following lemma is our final simplification statement. It improves over Lemma 5.2.14 in two respects. First, this lemma applies to any depth-$(d+1)$ read-once NAND formula, whether internally unpredictable or not. Second, this lemma brings $\Delta(f)$ all the way down to $\mathrm{polylog}\, n$. The penalties for these improvements are that we only get

sandwichers and we have to apply our pseudorandom restriction a few more times.

Lemma 5.2.16. *Let $1 \le d \le \log n$, let f be a depth-$(d+1)$ read-once* NAND *formula, and let $\varepsilon > 0$. For suitable values $\gamma = (\varepsilon/n)^{O(1)}$ and $t = O(d \cdot \log\log(n/\varepsilon) \cdot \log\log n)$, let (Y, Z) be a distribution over $\{0,1\}^{2n}$ that fools depth-d read-once $\mathbf{AC}^0$ formulas with error γ and sample $R \leftarrow \mathsf{Res}(Y,Z)^{\circ t}$. Then with probability $1-\varepsilon$, $f|_R$ can be ε-sandwiched between depth-$(d+1)$ read-once* NAND *formulas f_ℓ, f_u such that*

$$\Delta(f_\ell) \le O(\log^8(n/\varepsilon))$$
$$\Delta(f_u) \le O(\log^8(n/\varepsilon)).$$

Proof. For simplicity, we will only describe the upper sandwicher. Let $r = O(\log\log n)$, let $\delta = \Theta(\varepsilon/r)$, let $t_0 = O(d \log\log(n/\delta))$, and sample i.i.d. restrictions $R_0, R_1, \ldots, R_r \sim \mathsf{Res}(Y,Z)^{t_0}$. Let $f_0 = f$, and let f_{i+1} be the upper δ-sandwicher from applying Lemma 5.2.15 to $f_i|_{R_i}$, so f_{i+1} is γ-internally-unpredictable. Our upper sandwicher is $f_u = f_r$. Clearly, $f \le f_u$ and $\mathbb{E}[f_u - f] \le \delta \log\log n \le \varepsilon/2$. Now let us analyze $\Delta(f_u)$.

For each i, by Lemma 5.2.14, there is some constant c such that

$$\Pr_{R_i}[\Delta(f_i|_{R_i}) \le c\sqrt{\Delta(f_i)} \cdot \log^2(n/\varepsilon)] \ge 1 - \delta.$$

By the union bound, we may assume that this occurs for every i. Since Lemma 5.2.15 respects tree structure, we have $\Delta(f_{i+1}) \le \Delta(f_i|_{R_i})$, so $\Delta(f_{i+1}) \le c\sqrt{\Delta(f_i)} \cdot \log^2(n/\varepsilon)$.

If $\Delta(f_i) > c^4 \log^8(n/\varepsilon)$, this implies that $\Delta(f_{i+1}) \le \Delta(f_i)^{3/4}$. Initially, since f is read-once, $\Delta(f_0) \le n$. If we start at $\Delta = n$ and apply the map $\Delta \mapsto \Delta^{3/4}$ a total of $r = O(\log\log n)$ times, we reach $O(1)$. Therefore, for some $i \le r$, we must have $\Delta(f_i) \le c^4 \log^8(n/\varepsilon)$. When that occurs, for all subsequent $j > i$, we have $\Delta(f_j) \le c^4 \log^8(n/\varepsilon)$ as well. Therefore, $\Delta(f_u) = \Delta(f_r) \le c^4 \log^8(n/\varepsilon)$ as desired. □

5.2.3 Fooling Simplified Read-Once $\mathbf{AC}^0$ Formulas

At this point, we have shown that read-once $\mathbf{AC}^0$ formulas simplify *in some sense* under a pseudorandom restriction. What really matters, though, is whether they simplify *in the sense that we can fool* the restricted formulas using a short seed length. In this section, we will show that a PRG by Meka, Reingold, and Tal [MRT19] works for read-once formulas with a low value of $\Delta(f)$. Their PRG was designed for XORs of small arbitrary-order ROBPs:

Theorem 5.2.17 ([MRT19])**.** *For any n, w, ℓ, ε, there exists an explicit ε-PRG for functions $f\colon \{0,1\}^n \to \{\pm 1\}$ of the form $f(x) = \prod_{i=1}^{m} (-1)^{f_i(x)}$, where $f_1, \dots, f_m$ are width-w arbitrary-order ROBPs reading disjoint sets of at most ℓ variables, with seed length*

$$\log(n/\varepsilon) \cdot O(\log \ell + \log\log(n/\varepsilon))^{2w+2}.$$

Corollary 5.2.18. *For any $n, d, \Delta, \varepsilon$, there exists an explicit ε-PRG for depth-d read-once* NAND *formulas f with $\Delta(f) \le \Delta$ with seed length*

$$\log(n/\varepsilon) \cdot O(d \log \Delta + \log\log(n/\varepsilon))^{2d+2}.$$

Proof. By the Fourier expansion of the AND_m function,

$$\neg f(x) = \mathsf{AND}(f_1(x), \dots, f_m(x)) = \sum_{S \subseteq [m]} \frac{(-1)^{|S|}}{2^m} \prod_{i \in S} (-1)^{f_i(x)}.$$

The functions $f_1, \dots, f_m$ read disjoint sets of variables since f is read-once. Furthermore, each f_i can be computed by a width-d ROBP by Claim 5.2.3. Finally, since $\Delta(f) \le \Delta$, each f_i reads at most Δ^d variables. Therefore, each product is ε fooled by the PRG of

Theorem 5.2.17 with $w = d$ and $\ell = \Delta^d$, hence $\neg f$ is fooled by the same PRG with error

$$\sum_{S \subseteq [m]} \left| \frac{(-1)^{|S|}}{2^m} \right| \cdot \varepsilon = \varepsilon.$$

Fooling $\neg f$ is equivalent to fooling f itself. □

5.2.4 The Recursive PRG Construction

So far, we have shown that read-once $\mathbf{AC}^0$ simplifies under a pseudorandom restriction, and we have shown that there is a PRG for the simpler formulas with a short seed. To obtain a PRG, it is natural to try sampling that pseudorandom restriction and then using the PRG for the simpler formulas to fill in the $\star$ positions. That approach does not quite automatically work, because there is a danger that the pseudorandom restriction might *distort the expectation* of the formula. (Perhaps there is some formula with low expectation where the restriction of that formula tends to have high expectation, or vice versa.)

The good news is that using the prior work of Forbes and Kelley [FK18], we can slightly modify our pseudorandom restriction and ensure that the expectation is approximately preserved. Building on several prior works [RSV13; HLV18; CHRT18], by a beautiful Fourier-analytic argument, Forbes and Kelley showed that the expectation of any constant-width arbitrary-order ROBP is preserved under pseudorandom restrictions based on small-bias distributions.

Theorem 5.2.19 ([FK18])**.** *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a width-w arbitrary-order ROBP and let $\delta > 0$. For a suitable value $\beta = 2^{-O(\log(nw/\delta)\cdot w \cdot \log\log n)}$, let Y and Z be independent β-biased random variables distributed over $\{0,1\}^n$, and let U be an independent uniform random element of $\{0,1\}^n$. Then $|\mathbb{E}[f|_{\mathsf{Res}(Y,Z)}(U)] - \mathbb{E}[f]| \leq \delta$.*

(Theorem 5.2.19 follows from one of Forbes and Kelley's lemmas [FK18, Lemma 7.2] by

plugging in the Fourier tail bound of Chattopadhyay, Hatami, Reingold, and Tal [CHRT18] and using the standard fact that a β-biased distribution is automatically $(2\beta \cdot 2^{-k/2})$-almost k-wise independent for every k.)

Let ε be the desired error of the PRG, and let γ and t be the corresponding value from Lemma 5.2.16. Our modified pseudorandom restriction for depth-$(d+1)$ read-once $\mathbf{AC}^0$ is

$$R = \mathsf{Res}(G_d \oplus Y, G'_d \oplus Z),$$

where $G_d, G'_d \in \{0,1\}^n$ are independent n-bit strings that $(\gamma/2)$-fool depth-d read-once $\mathbf{AC}^0$ formulas, and Y and Z are independent β-biased n-bit strings, where β is the value from Theorem 5.2.19 with $w = d+2$ and $\delta = \varepsilon/t$. (In the base case, to fool depth-2 read-once $\mathbf{AC}^0$, we simply use the prior PRG by Gopalan, Meka, Reingold, Trevisan, and Vadhan [GMRTV12].) Our PRG for depth-$(d+1)$ read-once $\mathbf{AC}^0$ outputs a sample from the distribution $R^{\circ t} \circ G_{\mathrm{MRT}}$, where G_{MRT} is a sample from the ε-PRG of Corollary 5.2.18 for depth-$(d+1)$ formulas with $\Delta = O(\log^8(n/\varepsilon))$ as in the conclusion of Lemma 5.2.16. Let us prove the correctness of this PRG.

Claim 5.2.20. *For any depth-$(d+1)$ read-once $\mathbf{AC}^0$ formula f,*

$$|\,\mathbb{E}[f(R^{\circ t} \circ G_{\mathrm{MRT}})] - \mathbb{E}[f]| \le O(\varepsilon).$$

Proof. The class of depth-d read-once $\mathbf{AC}^0$ formulas is closed under shifts, so each of $G_d \oplus Y$ and $G'_d \oplus Z$ fools depth-d read-once $\mathbf{AC}^0$ formulas with error $\gamma/2$. Furthermore, the class of depth-d read-once $\mathbf{AC}^0$ formulas is closed under restrictions, so the concatenation $(G_d \oplus Y, G'_d \oplus Z)$ fools depth-d read-once $\mathbf{AC}^0$ formulas with error γ. Therefore, we may apply Lemma 5.2.16 to the restriction $R^{\circ t}$. Let E be the good event of Lemma 5.2.16. Define F_ℓ, F_u to be the sandwichers of Lemma 5.2.16 if E occurs and otherwise just the constant 0 and 1

functions respectively. (So F_ℓ and F_u are random variables that depend on the restriction $R^{\circ t}$ but that are independent of G_{MRT}.) Sample $U \in \{0,1\}^n$ uniformly at random. Then

$$\begin{aligned}
\mathbb{E}[f(R^{\circ t} \circ G_{\mathrm{MRT}})] &\leq \mathop{\mathbb{E}}_{R^{\circ t}}\left[\mathop{\mathbb{E}}_{G_{\mathrm{MRT}}}[F_u(G_{\mathrm{MRT}})]\right] && (f|_{R^{\circ t}} \leq F_u) \\
&\leq \mathop{\mathbb{E}}_{R^{\circ t}}\left[\mathop{\mathbb{E}}_{U}[F_u(U)] + \varepsilon\right] && (G_{\mathrm{MRT}} \text{ fools } F_u) \\
&\leq \mathop{\mathbb{E}}_{R^{\circ t}}\left[\mathop{\mathbb{E}}_{U}[F_u(U)] \mid E\right] + \Pr[\neg E] + \varepsilon \\
&\leq \mathop{\mathbb{E}}_{R^{\circ t}}\left[\mathop{\mathbb{E}}_{U}[f|_{R^{\circ t}}(U)] + \varepsilon \mid E\right] + \Pr[\neg E] + \varepsilon && (\text{Sandwiching}) \\
&\leq \mathbb{E}[f(R^{\circ t} \circ U)] + 2\Pr[\neg E] + 2\varepsilon \\
&\leq \mathbb{E}[f(R^{\circ t} \circ U)] + 4\varepsilon && (\text{Lemma 5.2.16.})
\end{aligned}$$

A perfectly analogous argument proves a corresponding lower bound, so

$$|\mathbb{E}[f(R^{\circ t} \circ G_{\mathrm{MRT}})] - \mathbb{E}[f(R^{\circ t} \circ U)]| \leq 4\varepsilon.$$

Now, the class of parity functions is closed under shifts. Therefore, each of $G_d \oplus Y$ and $G'_d \oplus Z$ is β-biased. Furthermore, f can be computed by an arbitrary-order width-$(w+2)$ ROBP by Claim 5.2.3. Therefore, we may apply Theorem 5.2.19 to conclude that $|\mathbb{E}[f|_R(U)] - \mathbb{E}[f]| \leq \delta$. The class of arbitrary-order width-$(w+2)$ ROBPs is closed under restriction, so inductively we get

$$|\mathbb{E}[f(R^{\circ t} \circ U)] - \mathbb{E}[f]| \leq t\delta = \varepsilon.$$

Therefore, by the triangle inequality,

$$|\mathbb{E}[f(R^{\circ t} \circ G_{\mathrm{MRT}})] - \mathbb{E}[f]| \leq 5\varepsilon. \qquad \square$$

Finally, let us verify the seed length of our PRG. For simplicity, we will only consider

the case that d is a constant. We will show that we achieve seed length

$$O(\log(n/\varepsilon) \cdot (\log\log(n/\varepsilon))^{2d+2})$$

for fooling depth-d read-once $\mathbf{AC}^0$ formulas. In the base case $d = 2$, the prior PRG by Gopalan et al. [GMRTV12] indeed has seed length $O(\log(n/\varepsilon) \cdot (\log\log(n/\varepsilon))^3)$. Now consider our recursive construction. Let s be the seed length of a $(\gamma/2)$-PRG for depth-d read-once $\mathbf{AC}^0$ and consider the seed length of fooling depth-$(d+1)$ read-once $\mathbf{AC}^0$. By Theorem 1.2.8, the seed length for sampling R is

$$\begin{aligned} 2s + O(\log(n/\beta)) &= 2s + O(\log(n/\delta)\log\log n) \\ &= 2s + O(\log(n/\varepsilon)\log\log n) \\ &= O(s). \end{aligned}$$

Therefore, the seed length for sampling $R^{\circ t}$ is $O(st) = O(s\log\log(n/\varepsilon)\log\log n)$, which is

$$O(\log(n/\varepsilon) \cdot (\log\log(n/\varepsilon))^{2(d+1)+2})$$

by induction. Meanwhile, the seed length for G_{MRT} is $O(\log(n/\varepsilon) \cdot (\log\log(n/\varepsilon))^{2(d+1)+2})$ as well, so the total seed length of our PRG is indeed $O(\log(n/\varepsilon) \cdot (\log\log(n/\varepsilon))^{2(d+1)+2})$.

When $d = \omega(1)$, a slightly more careful calculation [DHH19] shows that the seed length for fooling depth-d read-once $\mathbf{AC}^0$ is

$$\log(n/\varepsilon) \cdot O(d\log\log(n/\varepsilon))^{2d+2}.$$

5.3 PRGs for Constant-Depth Threshold Circuits

In this section, we study PRGs for models of computation based on (linear) *threshold functions*.

Definition 5.3.1. *A* threshold function *is a function* $\Phi\colon \{\pm 1\}^n \to \{\pm 1\}$ *of the form*

$$\Phi(x) = \operatorname{sign}(\langle w, x\rangle - \theta)$$

for some $w \in \mathbb{R}^n$ *and some* $\theta \in \mathbb{R}$.[3]

We have formulated the definition over the domain $\{\pm 1\}^n$ for convenience. Concepts that we previously formulated over $\{0,1\}^n$, such as PRGs and restrictions, can obviously be "translated" to the $\{\pm 1\}^n$ setting. For example, AND, OR, and MAJ are all threshold functions.

Meka and Zuckerman showed that any threshold function can be sandwiched by polynomial-width ROBPs, so obtaining an optimal PRG for threshold functions is necessary for obtaining an optimal PRG for polynomial-width ROBPs [MZ13]. Today, we indeed have explicit PRGs for threshold functions with near-optimal seed length $\widetilde{O}(\log(n/\varepsilon))$ [RS10; DGJSV10; MZ13; GKM18].

A natural next challenge is to fool models of computation based on combining *multiple* threshold functions. Recall that $\mathbf{TC}^0$ is the class of languages that can be computed by constant-depth polynomial-size *threshold circuits*, i.e., unbounded-fan-in circuits where each gate is an arbitrary threshold function. We will present a PRG for constant-depth threshold circuits of slightly superlinear size, where size is measured by counting the number of wires in the circuit. This PRG is joint work with Pooya Hatami, Avishay Tal, and Roei Tell [HHTT21].

[3] For simplicity, let us adopt the convention that $\operatorname{sign}(0) = +1$.

Theorem 5.3.2 (Joint with Pooya Hatami, Avishay Tal, and Roei Tell [HHTT21])**.** *For every $n, d \in \mathbb{N}$, there is a $\delta = 2^{-O(d)}$ and an explicit PRG for depth-d threshold circuits with $n^{1+\delta}$ wires with seed length $O(n^{1-\delta})$ and error $2^{-n^{\delta}}$.*

Ours is the first nontrivial PRG for this class, and furthermore it implies the first nontrivial algorithm for deterministically estimating the acceptance probability of a given circuit in this class. Admittedly, our seed length $O(n^{1-\delta})$ is only slightly nontrivial. Our excuse is the state of the art regarding lower bounds for $\mathbf{TC}^0$. Decades ago, Impagliazzo, Paturi, and Saks showed that depth-d threshold circuits computing the PARITY function on n bits must have at least $n^{1+2^{-O(d)}}$ wires [IPS97]. An explicit PRG with a significantly shorter seed than ours would imply a better lower bound for some function in **NP** (see Section 1.2.1), which is currently not known. Similarly, the error of our PRG essentially matches the currently-known average-case lower bounds for threshold circuits [CSS18], as we explain in more detail in our paper [HHTT21].

We allow our threshold circuits to read each variable multiple times. The reader might feel that we have strayed far from our main topic of ROBPs. There is some truth to this, but one can show that *read-once* $\mathbf{TC}^0$ is sandwiched by polynomial-width arbitrary-order ROBPs. Therefore, fooling read-once $\mathbf{TC}^0$ is a natural step toward designing better PRGs for ROBPs. The class of general $\mathbf{TC}^0$ circuits is more powerful, but in some sense it is a more "fundamental" class, so there is value in understanding what we can do with general $\mathbf{TC}^0$ circuits prior to trying to get a near-optimal seed length for read-once $\mathbf{TC}^0$. Additionally, we will see that one of the *ingredients* for our PRG for threshold circuits is a PRG for combinatorial rectangles, which can be computed by ROBPs. Thus, our PRG for threshold circuits is connected to **L** vs. **BPL** after all. Still, because this section is relatively far removed from ROBPs, we will give fewer details in our proofs; the curious reader is invited to read the paper [HHTT21].

For comparison, PRGs were known previously for near-quadratic-size depth-two thresh-

old circuits [ST17a] and for $\mathbf{AC}^0$ circuits with a few threshold gates [Vio07; LS11; ST18]. Meanwhile, for depth-d threshold circuits with $n^{1+2^{-O(d)}}$ wires, Tell previously gave a near-polynomial-time algorithm for the "quantified derandomization" problem [Tel18], which is incomparable with our PRG.

5.3.1 Simplification of Threshold Functions under Restrictions

Like our PRG for read-once $\mathbf{AC}^0$, our PRG for threshold circuits is based on pseudorandom restrictions. We will show that threshold circuits simplify in some sense under such restrictions, and then we will show how to fool the simplified functions.

We begin by studying the effect of a pseudorandom restriction on an individual threshold function, i.e., a single gate in our circuit. For context, prior to our work, Chen, Santhanam, and Srinivasan proved that a threshold function is likely to become close to a constant under a truly random restriction [CSS18].[4] In the following theorem statement, think of ε as being much smaller than p, e.g., $p = 1/n^{0.1}$ and $\varepsilon = 2^{-n^{0.001}}$.

Theorem 5.3.3 ([CSS18])**.** *Let $\Phi\colon \{\pm 1\}^n \to \{\pm 1\}$ be a threshold function. For any $p, \varepsilon > 0$,*

$$\Pr[\Phi|_{\mathcal{R}_p} \text{ is not } \varepsilon\text{-close to a constant}] \leq p^{\Omega(1)} \cdot \log(1/\varepsilon).$$

Intuition: Majority

We will need to prove a modified version of Theorem 5.3.3. To build intuition, let us first sketch the proof of Theorem 5.3.3 for the case of the majority function $\Phi(x) = \operatorname{sign}(\sum_i x_i)$. For simplicity, consider a truly random restriction $R \in \{+1, -1, \star\}^n$ that leaves *exactly* pn variables alive. By considering the asymptotics of the central binomial coefficients, we see

[4] Chen, Santhanam, and Srinivasan proved that except with failure probability $p^{\Omega(1)}$, the restricted function is $(2^{-p^{-\Omega(1)}})$-close to a constant [CSS18]. Theorem 5.3.3 follows immediately, because if ε is very small compared to p, then our claimed failure probability is larger than 1.

that for any $\Delta \geq 1$, we have the *anti-concentration bound*

$$\Pr_R \left[\left| \sum_{i \notin R^{-1}(\star)} R_i \right| \leq \Delta \right] \leq O\left(\frac{\Delta}{\sqrt{n}} \right). \tag{5.9}$$

Now sample $U \in \{\pm 1\}^{pn}$ uniformly at random to assign values to the remaining bits. By a standard *concentration bound* such as Hoeffding's inequality, it is unlikely that U overcomes the imbalance established by the restriction R:

$$\Pr_U \left[\left| \sum_{i \in R^{-1}(\star)} U_i \right| > \Delta \right] \leq \exp\left(-\Omega\left(\frac{\Delta^2}{pn} \right) \right). \tag{5.10}$$

Picking $\Delta = \Theta\left(\sqrt{pn \log(1/\varepsilon)} \right)$, we ensure that the two failure probabilities in Eq. (5.9) and Eq. (5.10) are $O\left(\sqrt{p \log(1/\varepsilon)} \right)$ and ε respectively. Thus, with decent probability, $\Phi|_R$ is extremely close to a constant.

Derandomized simplification with low failure probability

We would like to strengthen Theorem 5.3.3 in two respects. First of all, we want a *derandomized* version. Now that we are working over ± 1, we will think of Res as mapping $\{\pm 1\}^n \times \{0,1\}^n \to \{+1, -1, \star\}^n$. We will identify n-bit strings with subsets of $[n]$ in the natural way; we will often think of the second argument of Res as the *set* of $\star$ positions instead of the indicator vector for $\star$ positions. We would like to prove that threshold functions simplify under a restriction of the form $\mathsf{Res}(U, Z)$, where U (the assigned values) is still uniform random,[5] but Z (the set of $\star$ positions) is pseudorandom. We will take Z to be *k-wise independent with marginals p*, meaning that for each $i \in [n]$, $\Pr[i \in Z] = p$, and

[5] This is different than what we did for read-once $\mathbf{AC}^0$, where both components of the restriction were pseudorandom. Analyzing the case of uniform random assigned values will suffice by a trick due to Ajtai and Wigderson [AW89].

these events are k-wise independent. A derandomized version of Theorem 5.3.3 was indeed proven previously by Tell [Tel18].

Second of all, we shall consider two different types of failures: bad assigned bits (U) or bad stars (Z). We are willing to tolerate a moderate probability of bad assigned bits, similar to the $p^{\Omega(1)} \cdot \log(1/\varepsilon)$ bound in Theorem 5.3.3, but we would like the probability of bad stars to be much lower (at most ε). Unfortunately, even under a truly random restriction, the probability of bad stars genuinely is not that low. (For example, if f is a dictator function, then the probability of bad stars is clearly p.) To evade this obstacle, we shall settle for a modified, "slightly non-black-box" restriction. We will assign values to a pseudorandom subset of the variables *and* to a few additional variables, namely the m variables with the largest coefficients in absolute value. The precise statement follows. Unfortunately, it is fairly technical; for simplicity of notation we reindex variables so that the weight vector is sorted.

Lemma 5.3.4. *Let $\Phi(x) = \operatorname{sign}(\langle w, x\rangle - \theta)$ be a threshold function with $|w_1| \geq \cdots \geq |w_n|$. Let $p, \varepsilon, \gamma \in (0, 1/2)$. Sample $U \in \{\pm 1\}^n$ uniformly at random and sample $Z \subseteq [n]$ from a $\log(1/\varepsilon)$-wise independent distribution with marginals p. For a suitable value*

$$m = O(\gamma^{-2} \cdot \log^2(1/\varepsilon) \cdot \log^2(\log(1/\varepsilon)/\gamma)) = \widetilde{O}(\gamma^{-2} \cdot \log^2(1/\varepsilon)),$$

let $T \subseteq [n]$ be a fixed set with $T \cap [m] = \varnothing$. Then

$$\Pr_Z \left[\Pr_U [\Phi|_{\mathsf{Res}(U, Z \cap T)} \text{ is not } \varepsilon\text{-close to a constant}] \leq \gamma + O\left(\sqrt{p \log(1/\varepsilon)}\right) \right] \geq 1 - \varepsilon.$$

(The letter T stands for "tail.") At a high level, the proof of Lemma 5.3.4 uses the same essential steps that we used in our discussion of the majority function: an anti-concentration bound followed by a concentration bound. For simplicity, we will present a

proof of Lemma 5.3.4 with

$$m = O(\gamma^{-4} \cdot \log^3(1/\varepsilon)),$$

which is weaker than the bound claimed in Lemma 5.3.4 but still sufficient for proving Theorem 5.3.2 (the existence of explicit nontrivial PRGs for threshold circuits). See the paper [HHTT21] for the more refined proof of Lemma 5.3.4 that achieves the smaller value of m.

High-probability anti-concentration

Toward proving Lemma 5.3.4, let $I = Z \cap T$ (the set of living variables), and let $J = [n] \setminus I$ (the set of variables that are assigned values). Our anti-concentration bound is as follows.

Claim 5.3.5 (Anti-concentration)**.** *With probability at least $1 - \varepsilon$ over Z, we have*

$$\Pr_U \left[|\langle w_J, U_J \rangle - \theta| \le \sqrt{2\log(2/\varepsilon)} \cdot \|w_I\| \right] \le \gamma + O\left(\sqrt{p \log(1/\varepsilon)}\right). \tag{5.11}$$

The proof of Claim 5.3.5 is split into two cases depending on whether the tail w_T is "regular."

Definition 5.3.6. *We say $w \in \mathbb{R}^n$ is* μ-regular *if for every i, we have $|w_i| \le \mu \|w\|$.*

The regular case

We begin with the case that w_T is regular. We rely on the classic Berry-Esseen theorem, a bound on the convergence rate in the central limit theorem.

Theorem 5.3.7 (A corollary of the Berry-Esseen theorem [DGJSV10, Corollary 2.4])**.** *Let $w \in \mathbb{R}^n$ be μ-regular. Sample $U \in \{\pm 1\}^n$ uniformly at random. Then for every $\theta \in \mathbb{R}$ and every $\Delta > 0$,*

$$\Pr[|\langle w, U \rangle - \theta| \le \Delta \cdot \|w\|] \le 2 \cdot (\Delta + \mu).$$

We will also use a tail bound for k-wise independent events. We state here a special case of a result by Bellare and Rompel [BR94].

Theorem 5.3.8 ([BR94, Lemma 2.3])**.** *Let $X_1, \dots, X_n \in [0,1]$ be k-wise independent, where $k \geq 4$. Let $\mu = \sum_i \mathbb{E}[X_i]$, and let Δ satisfy $\Delta \geq \mu/2$ and $\Delta \geq 300k$. Then*

$$\Pr\left[\left|\mu - \sum_i X_i\right| \geq \Delta\right] \leq 2^{-k}.$$

For the remainder of this section, fix a suitable value $\mu = \Theta(\gamma/\log(1/\varepsilon))$.

Proof of Claim 5.3.5 assuming w_T is μ-regular. For $i \in T$, let

$$X_i = \frac{w_i^2}{\mu^2 \|w_T\|^2} \cdot \mathbf{1}[i \in I],$$

so $X_i \in [0,1]$ and $\mathbb{E}[X_i] = \frac{pw_i^2}{\mu^2\|w_T\|^2}$. By tail bounds for k-wise independence (Theorem 5.3.8),

$$\Pr_Z\left[\sum_i X_i \leq O(p/\mu^2 + \log(1/\varepsilon))\right] \geq 1 - \varepsilon.$$

Therefore, there is a value $\lambda = O(p + \mu^2 \log(1/\varepsilon))$ such that

$$\Pr_Z[\|w_I\|^2 \leq \lambda \|w_T\|^2] \geq 1 - \varepsilon.$$

Fix any Z for which that good event occurs. We may assume without loss of generality that $\lambda \leq 3/4$, because otherwise the claimed failure probability $\gamma + O\left(\sqrt{p\log(1/\varepsilon)}\right)$ is bigger than 1. Therefore,

$$\|w_{J\cap T}\|^2 = \|w_T\|^2 - \|w_I\|^2 \geq \|w_T\|^2/4.$$

Consequently, $w_{J\cap T}$ is (2μ)-regular, so for any $\theta' \in \mathbb{R}$ and any $\Delta > 0$, by the Berry-Esseen

theorem (Theorem 5.3.7),

$$\Pr_U[|\langle w_{J\cap T}, U_{J\cap T}\rangle - \theta'| \le \Delta \cdot \|w_I\|] \le \Pr_U[|\langle w_{J\cap T}, U_{J\cap T}\rangle - \theta'| \le 2\sqrt{\lambda} \cdot \Delta \cdot \|w_{J\cap T}\|] \le O(\Delta\sqrt{\lambda} + \mu).$$

By considering picking $U_{J\setminus T}$ first, then $U_{J\cap T}$ second, this implies that

$$\Pr_U[|\langle w_J, U_J\rangle - \theta| \le \Delta \cdot \|w_I\|] \le O(\Delta\sqrt{\lambda} + \mu).$$

Picking $\Delta = \sqrt{2\log(2/\varepsilon)}$, the failure probability comes out to $\gamma + O\left(\sqrt{p\log(1/\varepsilon)}\right)$ due to our choice of μ. □

The irregular case

Now we present the proof of Claim 5.3.5 in the case that w_T is irregular. We rely on a famous result by Littlewood and Offord [LO43], subsequently improved by Erdős [Erd45].

Theorem 5.3.9 ([LO43; Erd45])**.** *Let $w \in \mathbb{R}^n$ and assume that for every i, we have $|w_i| \ge 1$. Sample $U \in \{\pm 1\}^n$ uniformly at random. Then for every $\theta \in \mathbb{R}$ and every $\Delta \ge 1$,*

$$\Pr[|\langle w, U\rangle - \theta| \le \Delta] \le O(\Delta/\sqrt{n}).$$

Proof of Claim 5.3.5 assuming w_T is not μ-regular. In this case, we will achieve failure probability *zero* with respect to Z and failure probability γ with respect to U. By Littlewood-Offord-Erdős (Theorem 5.3.9), for any $\theta' \in \mathbb{R}$ and any $\Delta \ge \mu \cdot \|w_T\|$, we have

$$\Pr_U[|\langle w_{[m]}, U_{[m]}\rangle - \theta'| \le \Delta] \le O\left(\frac{\Delta}{\mu \cdot \|w_T\| \cdot \sqrt{m}}\right).$$

(After all, every coordinate of $\frac{1}{\mu \cdot \|w_T\|} \cdot w_{[m]}$ is at least 1 in absolute value.) Therefore, for any fixing of Z, we have

$$\Pr_U[|\langle w_J, U_J\rangle - \theta| \le \Delta] \le O\left(\frac{\Delta}{\mu \cdot \sqrt{m} \cdot \|w_T\|}\right).$$

(Imagine picking $U_{J\setminus[m]}$ first, then $U_{[m]}$ second.) Setting $\Delta = \sqrt{2\log(2/\varepsilon)} \cdot \|w_T\| \gg \mu \cdot \|w_T\|$, we get

$$\Pr_U\left[|\langle w_J, U_J\rangle - \theta| \le \sqrt{2\log(2/\varepsilon)}\right] \le O\left(\frac{\sqrt{\log(1/\varepsilon)}}{\mu \cdot \sqrt{m}}\right) \le \gamma$$

by our choices of μ and m. □

Concentration for the restricted function

At this point, we have completed the proof of our anti-concentration bound (Claim 5.3.5). To wrap up the proof of our simplification lemma for threshold functions (Lemma 5.3.4), we will combine our anti-concentration bound with a standard concentration bound to show that the restricted function is close to a constant, just like we did in our warm-up analysis of the simple majority function.

Proof of Lemma 5.3.4. Let $\Phi' = \Phi|_{\mathsf{Res}(U, Z\cap T)}$. Then

$$\Phi'(x) = \operatorname{sign}(\langle w_I, x_I\rangle - \theta'),$$

where $\theta' = \theta - \langle w_J, U_J\rangle$. By Claim 5.3.5, we may assume that $|\theta'| > \sqrt{2\log(2/\varepsilon)} \cdot \|w_I\|$. In this case, by Hoeffding's inequality, Φ' is ε-close to a constant. □

5.3.2 Using Queries to Eliminate One Layer of the Circuit

In the last section, we showed that threshold functions become close to a constant under a certain type of pseudorandom restriction. We would like to show that threshold circuits simplify under pseudorandom restrictions, but we face two challenges. The first challenge is that the pseudorandom restrictions we analyzed in the last section are "slightly non-black-box," i.e., we forced the restriction to assign values to the variables with large weights in Lemma 5.3.4. The second challenge is that the failure probability in Lemma 5.3.4 is too large. In particular, it is larger than p. (Looking ahead, we will need the simplification failure probability to be less than p to get any nontrivial PRG, because we will use the Ajtai-Wigderson framework [AW89], which involves a union bound over roughly p^{-1} applications of the pseudorandom restriction.)

Our solution to both of these challenges is a carefully engineered model of computation that we will use to formulate our notion of "simplification." Roughly speaking, a *threshold decision tree* is a decision tree that can query both variables and threshold functions with threshold circuits at the leaves. The precise definition follows.

Definition 5.3.10 (Threshold decision tree)**.** *A (d, w, D, M, e)-*threshold decision tree *is a full binary tree $\mathcal{T}$. Each internal node is labeled with either a variable or a threshold function, and the two edges from the node to its children are labeled $+1$ and -1. We require that on each root-to-leaf path, there are at most D nodes labeled with variables and at most M nodes labeled with threshold functions. Each leaf ℓ is labeled with with a depth-d threshold circuit C_ℓ with at most w wires. The leaf ℓ is additionally labeled with a function $\mathsf{Err}_\ell \colon \{\pm 1\}^n \to \{0, 1, \ldots, e\}$ of the form $\mathsf{Err}_\ell(x) = \sum_{i=1}^{e} \mathbf{1}[\Phi_{\ell,i}(x) = -1]$, where $\Phi_{\ell,1}, \ldots, \Phi_{\ell,e}$ are threshold functions.*

We identify $\mathcal{T}$ with a function $\mathcal{T} \colon \{\pm 1\}^n \to \{\pm 1\}$ defined as follows. On input $x \in \{\pm 1\}^n$, we start at the root node. When we reach an internal node labeled by a function Φ (either a threshold function or an individual variable), we evaluate $\Phi(x)$ to decide which

child to move to. Finally, when we reach a leaf $\ell = \ell(x)$, we set $\mathcal{T}(x) = C_\ell(x)$. We also set $\mathsf{Err}(\mathcal{T}, x) = \mathsf{Err}_\ell(x) \in \{0, 1, \ldots, e\}$, and we define $\mathsf{Err}(\mathcal{T}) = \mathbb{E}[\mathsf{Err}(\mathcal{T}, U_n)]$. Finally, we say that a function $f\colon \{\pm 1\}^n \to \{\pm 1\}$ is consistent with *$\mathcal{T}$ if for every $x \in \{\pm 1\}^n$, if $\mathsf{Err}(\mathcal{T}, x) = 0$, then $\mathcal{T}(x) = f(x)$.*

Intuitively, each $\Phi_{\ell,i}$ function is a flag saying that something went wrong and the computation should not be trusted. The function $\mathsf{Err}(\mathcal{T}, x)$ counts the number of flags that are raised in the computation of $\mathcal{T}(x)$. We will construct trees such that $\mathsf{Err}(\mathcal{T})$, the expected number of error flags on a uniform random input, is close to zero.

Chen, Santhanam, and Srinivasan studied a similar model of computation [CSS18]. Other "hybrid decision tree" models of computation have found applications elsewhere in pseudorandomness and circuit complexity [Hås14; ST17a; Tal17; ST19a; HHTT21].

We will prove that if we start with a depth-d threshold circuit with w wires where w is slightly superlinear and we hit it with a suitable pseudorandom restriction, the restricted function is consistent with a $(d-1, w, D, M, e)$-threshold decision tree with nontrivial parameters D, M, e. We make progress in the sense that we go from depth-d circuits to depth-$(d-1)$ circuits. The main penalty we pay is all the queries to variables and threshold functions. We begin with the case that every variable in the circuit has bounded fan-out.

Lemma 5.3.11 (Depth reduction for threshold circuits with bounded variable-fan-out)**.** *Let f be a depth-d threshold circuit with at most w wires where every variable has fan-out at most t. Let $p, \varepsilon > 0$, let $k = p^{-5/6} \log(6w/\varepsilon)$, and let $R = \mathsf{Res}(U, Z)$, where U is uniform random and Z is k-wise independent with marginals p. Then $f|_R$ is consistent with a $(d-1, w, D, M, w)$-threshold decision tree $\mathcal{T}$ such that $\mathbb{E}[\mathsf{Err}(\mathcal{T})] \leq \varepsilon$ and*

$$D \leq O((p^{7/6} \cdot w + p^{-5/2} \cdot t) \cdot \log^2(w/\varepsilon))$$
$$M \leq O(p^{-2/3} \cdot t).$$

To be clear, in the statement of Lemma 5.3.11, $\mathcal{T}$ is a random variable, jointly distributed with the restriction R, which is why it makes sense to speak of $\mathbb{E}[\mathsf{Err}(\mathcal{T})]$. In the complexity bounds for $\mathcal{T}$, the most important term is the $p^{7/6} \cdot w$ term in the bound on D. Note that R will leave roughly $p \cdot n$ variables alive. A decision tree that makes $D \approx p \cdot n$ queries would therefore be all-powerful, i.e., it could compute any Boolean function on the $\approx p \cdot n$ living variables whatsoever. Since $7/6 > 1$ and we are thinking of $w \approx n$ and $p = n^{-\Omega(1)}$, we will have $p^{7/6} \cdot w = o(p \cdot n)$, i.e., $\mathcal{T}$ has *slightly sub-maximal* depth. The bound $p^{7/6} \cdot w$ is reminiscent of *shrinkage exponents* [Sub61; And87; PZ93; IN93; DZ94; HRY95; Hås98; Tal14; IMZ19], but note that D and w are two different complexity measures (number of variable queries vs. number of wires).

Toward proving Lemma 5.3.11, let $\mathcal{G}$ be the bottom layer[6] of gates in f. For each gate $\Phi \in \mathcal{G}$, let fan-in(Φ) denote the number of variables feeding into Φ. We partition $\mathcal{G} = \mathcal{G}^+ \cup \mathcal{G}^-$, where $\mathcal{G}^+$ (the "heavy gates") consists of gates with fan-in(Φ) $\geq p^{-5/6}$ and $\mathcal{G}^-$ (the "light gates") consists of gates with fan-in(Φ) $< p^{-5/6}$. We tackle these two sets in turn.

Heavy gates

At a high level, our hope is that the heavy gates will become extremely biased, so we can safely replace them with constants. This hope will not always be realized; we will use queries to handle the gates that remain somewhat balanced.

Since each heavy gate has fan-in at least $p^{-5/6}$, the number of heavy gates is bounded by $|\mathcal{G}^+| \leq p^{5/6} \cdot w$. (This is actually the only way we use the high fan-in of the heavy gates.)

Define $\gamma = p^{1/3} \log(w/\varepsilon)$, and let m be the value from Lemma 5.3.4 with error parameter $\varepsilon/(6w)$ rather than ε, so $m = O(p^{-2/3} \log^2(1/p)) = \widetilde{O}(p^{-2/3})$. For each gate $\Phi \in \mathcal{G}^+$,

[6] More precisely, $\mathcal{G}$ is the set of gates Φ such that every input to Φ is a variable rather than another gate. We do not need to assume that f is literally layered.

let H_Φ be the set of the m (indices of) variables feeding into Φ with the largest weights in absolute value. Let $H = \bigcup_{\Phi \in \mathcal{G}^+} H_\Phi$ (H for "head"), and let $T = [n] \setminus H$ (the "tail").

Lemma 5.3.12. *With probability at least $1-\varepsilon/6$ over the choice of Z, for every gate $\Phi \in \mathcal{G}^+$, we have*

$$\Pr_U[\Phi|_{\mathsf{Res}(U,Z\cap T)} \text{ is not } (\varepsilon/(6w))\text{-close to a constant}] \leq O(p^{1/3}\log(w/\varepsilon)).$$

Proof. This follows from Lemma 5.3.4 and a union bound over the $\leq w$ gates in $\mathcal{G}^+$. □

Lemma 5.3.12 is dissatisfying in two respects, in keeping with our discussion prior to Definition 5.3.10. First, the restriction $\mathsf{Res}(U, Z \cap T)$ has fewer stars than the restriction $R = \mathsf{Res}(U, Z)$ that we are supposed to be analyzing. We will solve this problem by querying the necessary variables in H; the next claim bounds the number of queries.

Claim 5.3.13. *With probability at least $1 - \varepsilon/6$ over the choice of Z, we have*

$$|Z \cap H| \leq O(p^{7/6} \cdot w \cdot \log^2(1/p) + \log(1/\varepsilon)).$$

Proof. Clearly,

$$|H| \leq m \cdot |\mathcal{G}_+| \leq O(p^{-2/3}\log^2(1/p) \cdot p^{5/6}w) = O(p^{1/6} \cdot w \cdot \log^2(1/p)).$$

The claim follows by Theorem 5.3.8. □

The second dissatisfying aspect of Lemma 5.3.12 is that the failure probability with respect to U is only bounded by $O(p^{1/3}\log(w/\varepsilon))$, which is much larger than ε. We will solve this problem using the *read-t Chernoff bound* [GLSS15].

Theorem 5.3.14 (Read-t Chernoff bound [GLSS15])**.** *Let $U^{(1)}, \ldots, U^{(n)}$ be independent random variables. Let $Q_1, \ldots, Q_m$ be $\{0,1\}$-valued random variables, where each Q_i is a function of $U^{(1)}, \ldots, U^{(n)}$, and each $U^{(j)}$ influences at most t of the Q_i variables. Then for any $\alpha > 0$,*

$$\Pr\left[\sum_{i=1}^{m} Q_i \geq \left(\sum_{i=1}^{m} \mathbb{E}[Q_i]\right) + \alpha m\right] \leq e^{-2\alpha^2 m/t}.$$

Briefly, since each variable has fan-out at most t, the read-t Chernoff bound will imply that with high probability, the *fraction* of gates that simplify is roughly $O(p^{1/3}\log(w/\varepsilon))$. We will query the variables feeding into the gates that fail to simplify (or in a few cases, the gates themselves). The next two claims prove that this idea works and bounds the number of queries.

Claim 5.3.15. *With probability at least $1-\varepsilon/6$ over the choice of Z, for every gate $\Phi \in \mathcal{G}^+$, the number of living variables feeding into Φ is at most $O(p \cdot \text{fan-in}(\Phi) + \log(w/\varepsilon))$.*

Proof. This follows from Theorem 5.3.8 and the union bound. □

Claim 5.3.16. *Let Z be such that the conclusions of Lemma 5.3.12 and Claim 5.3.15 both hold. With probability $1-\varepsilon/6$ over U, $\mathcal{G}^+$ can be partitioned into three parts, $\mathcal{G}^+ = \mathcal{G}_1^+ \cup \mathcal{G}_2^+ \cup \mathcal{G}_3^+$, such that under the restriction $\mathsf{Res}(U, Z \cap T)$, the following holds.*

1. *Every gate in $\mathcal{G}_1^+$ is $(\varepsilon/(6w))$-close to a constant.*
2. *The number of living wires feeding into $\mathcal{G}_2^+$ is $O(p^{7/6} \cdot w \cdot \log^2(w/\varepsilon))$.*
3. *$|\mathcal{G}_3^+| \leq O(p^{-2/3} \cdot t)$.*

Proof. We first partition the gates in $\mathcal{G}^+$ according to their fan-in. Let S_i be the set of gates in $\mathcal{G}^+$ with fan-in in the interval $(2^{i-1}, 2^i]$, so $\mathcal{G}^+ = S_1 \cup \cdots \cup S_{\log w}$. Let $\varphi = O(p^{1/3}\log(w/\varepsilon))$ be the failure probability in Lemma 5.3.12. Define $\mathcal{G}_3^+$ to be the union of all the parts S_i

where $|S_i| \leq \varphi^{-2} \cdot t \cdot \log(6(\log w)/\varepsilon)$. Define $\mathcal{G}_1^+$ to be the set of gates in $\mathcal{G}^+ \setminus \mathcal{G}_3^+$ that are $(\varepsilon/(6w))$-close to a constant after the restriction. Define $\mathcal{G}_2^+ = \mathcal{G}^+ \setminus (\mathcal{G}_1^+ \cup \mathcal{G}_3^+)$.

Clearly,

$$|\mathcal{G}_3^+| \leq \varphi^{-2} \cdot t \cdot \log(6(\log w)/\varepsilon) \cdot \log w \leq O(p^{-2/3} \cdot t).$$

Now we must bound the number of living wires feeding into $\mathcal{G}_2^+$. Consider, therefore, a part S_i with $|S_i| > \varphi^{-2} \cdot t \cdot \log(6(\log w)/\varepsilon)$. Since U is uniform random and each variable has fan-out t, we may apply the read-t Chernoff bound (Theorem 5.3.14) to find

$$\Pr\left[|S_i \cap \mathcal{G}_2^+| \geq 2\varphi|S_i|\right] \leq e^{-\varphi^2|S_i|/t} \leq \frac{\varepsilon}{6 \log w}.$$

Therefore, by the union bound, we may assume that for every i, we have $|S_i \cap \mathcal{G}_2^+| \leq 2\varphi|S_i|$. The number of living variables feeding into each gate in S_i is at most $O(p \cdot 2^i + \log(w/\varepsilon))$. Therefore, the number of living variables feeding into $\mathcal{G}_2^+$ is at most

$$\begin{aligned}\sum_{i=1}^{\log w} \cdot O(\varphi|S_i| \cdot p \cdot 2^i + \varphi|S_i| \cdot \log(w/\varepsilon)) &\leq O\left(p \cdot \varphi \cdot \sum_{i=1}^{\log w} |S_i| \cdot 2^i\right) + O(\varphi|\mathcal{G}^+| \cdot \log(w/\varepsilon)) \\ &\leq O(p \cdot \varphi \cdot w + \varphi \cdot p^{5/6} \cdot w \cdot \log(w/\varepsilon)) \\ &= O(p^{7/6} \cdot w \cdot \log^2(w/\varepsilon)).\end{aligned}$$

□

Light gates

At a high level, when we apply the restriction to a light gate, our hope is that it only has at most one surviving input variable, so we can replace the gate with a constant or a literal. Once again, this hope will not always be realized; we will query the variables feeding into gates that have multiple surviving inputs.

Define a *connected pair* to be an (unordered) pair of distinct variables that both feed into the same light gate. We say that a connected pair *survives* the restriction $R = \mathsf{Res}(U, Z)$

if both variables are assigned $\star$; note that this depends only on Z. The following lemma bounds the number of queries we will make to handle the light gates.

Lemma 5.3.17. *With probability at least $1 - \varepsilon/6$ over the choice of Z, the number of connected pairs that survive the restriction is at most $O(p^{7/6} \cdot w + p^{-5/2} \cdot t \cdot \log(w/\varepsilon))$.*

For intuition, let us first compute the *expected* number of connected pairs that survive the restriction. Since each light gate has fan-in at most $p^{-5/6}$, each wire participates in at most $p^{-5/6}$ connected pairs, hence the total number of connected pairs is at most $p^{-5/6} \cdot w$. Each connected pair survives the restriction with probability p^2, so the expected number of surviving connected pairs is at most $p^2 \cdot p^{-5/6} \cdot w = p^{7/6} \cdot w$. To prove that a similar bound holds with high probability, we will look for many gates that read disjoint sets of variables and apply tail bounds for k-wise independence; the details follow.

Proof of Lemma 5.3.17. Like in the proof of Claim 5.3.16, we partition $\mathcal{G}^- = S_1 \cup \cdots \cup S_r$, where S_i is the set of gates with fan-in in the interval $(2^{i-1}, 2^i]$, and $r = \log(p^{-5/6})$. For each i, define a graph $H_i = (S_i, E_i)$, where we connect two gates with an edge if they share an input variable. This graph H_i has maximum degree less than $p^{-5/6} \cdot t$, so it admits a proper coloring using $O(p^{-5/6} \cdot t)$ colors. Let the color classes of H_i be $\Omega_{i,1}, \ldots, \Omega_{i,q}$.

For each gate Φ, let Q_Φ denote the fraction of connected pairs feeding into Φ that survive the restriction, so $\mathbb{E}[Q_\Phi] = p^2$. For each i, j, since the coordinates of Z are k-wise independent for $k = p^{-5/6} \log(6w/\varepsilon)$, the variables Q_Φ for $\Phi \in \Omega_{i,j}$ are $\log(6w/\varepsilon)$-wise independent (note that gates in $\Omega_{i,j}$ do not share variables). Therefore, by Theorem 5.3.8,

$$\Pr\left[\sum_{\Phi \in \Omega_{i,j}} Q_\Phi \leq O(p^2|\Omega_{i,j}| + \log(w/\varepsilon))\right] \geq 1 - \frac{\varepsilon}{6w}.$$

There are at most w gates in total, so the total number of color classes $\Omega_{i,j}$ is at most w.

Therefore, by the union bound, we may assume that for every i, j simultaneously,

$$\sum_{\Phi \in \Omega_{i,j}} Q_\Phi \leq O(p^2 |\Omega_{i,j}| + \log(w/\varepsilon)).$$

As a consequence, for each set S_i, we have

$$\sum_{\Phi \in S_i} Q_\Phi \leq O(p^2 |S_i| + p^{-5/6} \cdot t \cdot \log(w/\varepsilon)).$$

Therefore, the number of connected pairs feeding into S_i that survive the restriction is given by

$$\begin{aligned}
\sum_{\Phi \in S_i} Q_\Phi \cdot \binom{\text{fan-in}(\Phi)}{2} &\leq \binom{2^i}{2} \cdot \sum_{\Phi \in S_i} Q_\Phi \\
&\leq \binom{2^i}{2} \cdot O(p^2 |S_i| + p^{-5/6} \cdot t \cdot \log(w/\varepsilon)) \\
&\leq O\left(p^2 \sum_{\Phi \in S_i} \binom{\text{fan-in}(\Phi)}{2} + \binom{2^i}{2} \cdot p^{-5/6} \cdot t \cdot \log(w/\varepsilon) \right).
\end{aligned}$$

Therefore, the total number of connected pairs that survive the restriction is at most

$$\begin{aligned}
&O\left(p^2 \cdot \sum_{\Phi \in \mathcal{G}^-} \binom{\text{fan-in}(\Phi)}{2} + p^{-5/6} \cdot t \cdot \log(w/\varepsilon) \cdot \sum_{i=1}^{r} \binom{2^i}{2} \right) \\
&\quad \leq O(p^2 \cdot w \cdot p^{-5/6} + p^{-5/6} \cdot t \cdot \log(w/\varepsilon) \cdot 2^{2r}) \\
&= O(p^{7/6} \cdot w + p^{-5/2} \cdot t \cdot \log(w/\varepsilon)).
\end{aligned}$$

□

The decision tree construction

With probability $1 - 4\varepsilon/6$ over the choice of Z, the conclusions of Lemmas 5.3.12 and 5.3.17 and Claims 5.3.13 and 5.3.15 are all satisfied. If this does not occur, we set $\mathcal{T}$ to be a trivial

tree with $\mathcal{T}(x) \equiv 0$ and $\mathsf{Err}(\mathcal{T}, x) \equiv 1$; assume now that it does occur.

On input $u \in \{0,1\}^Z$, our tree $\mathcal{T}$ queries every variable in $Z \cap H$. Together with the restriction R, this defines a restriction R' with $(R')^{-1}(\star) = Z \cap T$. If R' does not satisfy the conclusion of Claim 5.3.16, the tree halts; the corresponding leaf ℓ is labeled with $C_\ell \equiv 0$ and $\mathsf{Err}_\ell \equiv 1$. Assume now that the conclusion of Claim 5.3.16 is satisfied, so we get a partition $\mathcal{G}^+ = \mathcal{G}_1^+ \cup \mathcal{G}_2^+ \cup \mathcal{G}_3^+$.

Write $\mathcal{G}_1^+ = \{\Phi_1, \ldots, \Phi_r\}$ and note that $r \leq w$. Under R', each gate $\Phi_i \in \mathcal{G}_1^+$ is $(\varepsilon/(6w))$-close to a constant b_i. Define

$$\mathsf{Err}_i(u) = \mathbf{1}[\Phi_i|_{R'}(u) \neq b_i].$$

Our tree $\mathcal{T}$ queries all the living variables feeding into $\mathcal{G}_2^+$ and all the gates in $\mathcal{G}_3^+$. Finally, the tree queries all the variables in connected pairs, reaching some leaf ℓ. We label this leaf with the circuit C_ℓ obtained from f by (a) applying the restriction R, (b) replacing each variable queried with its value, and (c) replacing each gate $\Phi_i \in \mathcal{G}_1^+$ with the corresponding constant b_i. We furthermore set $\mathsf{Err}_\ell = \sum_{i=1}^r \mathsf{Err}_i$.

The circuit C_ℓ has depth $d-1$, because for every gate $\Phi \in \mathcal{G}$, either Φ was replaced with a constant, or else all but perhaps one of the inputs of Φ was replaced with a constant. The number of variable queries D made by the tree satisfies

$$\begin{aligned} D &\leq O(p^{7/6} \cdot w \cdot \log^2(1/p) + \log(1/\varepsilon)) \\ &\quad + O(p^{7/6} \cdot w \cdot \log^2(w/\varepsilon)) \\ &\quad + O(p^{7/6} \cdot w + p^{-5/2} \cdot t \cdot \log(w/\varepsilon)) \\ &\leq O(p^{7/6} \cdot w \cdot \log^2(w/\varepsilon) + p^{-5/2} \cdot t \cdot \log(w/\varepsilon)), \end{aligned}$$

where we assumed without loss of generality that $p \geq 1/w$, since otherwise the claimed

bound on D is more than w, which can be achieved trivially. Meanwhile, the number of threshold function queries M made by the tree satisfies $M = |\mathcal{G}_3^+| \leq O(p^{-2/3} \cdot t)$.

All that remains is to bound $\mathbb{E}[\mathsf{Err}(\mathcal{T})]$. Unpacking the definitions, we have

$$\mathbb{E}[\mathsf{Err}(\mathcal{T})] = \mathop{\mathbb{E}}_{U,Z}[\mathsf{Err}(\mathcal{T}, U_Z)].$$

There is at most a $4\varepsilon/6$ chance that we get a bad restriction R that causes our tree to halt early; this contributes $4\varepsilon/6$ to $\mathbb{E}[\mathsf{Err}(\mathcal{T})]$. Furthermore, we have $R' = \mathsf{Res}(U, Z \cap T)$, so the probability that R' fails to satisfy the conclusion of Claim 5.3.16 is at most $\varepsilon/6$. Finally, for each $i \in [r]$, remember that we chose b_i so that

$$\Pr[\Phi_i|_{R'}(U_{Z \cap H}) \neq b_i] \leq \frac{\varepsilon}{6w},$$

so the expected number of Err_i that output 1 is at most $\varepsilon/6$. Therefore, overall we get $\mathbb{E}[\mathsf{Err}(\mathcal{T})] \leq \varepsilon$ as claimed. This completes the proof of Lemma 5.3.11.

Removing the assumption of bounded fan-out

In Lemma 5.3.11, we assumed that each variable has a fan-out of at most t. We now consider general threshold circuits, with no bound on the variable fan-out. We will perform a simple reduction to the case of bounded fan-out by querying all the high-fan-out variables.

Lemma 5.3.18 (Depth reduction for threshold circuits)**.** *Let f be a depth-d threshold circuit with at most w wires. Let $p, \varepsilon > 0$, let $k = p^{-5/6} \log(6w/\varepsilon)$, and let $R = \mathsf{Res}(U, Z)$, where U is uniform random and Z is k-wise independent with marginals p. Then $f|_R$ is consistent*

with a $(d-1, w, D, M, w)$*-threshold decision tree* $\mathcal{T}$ *such that* $\mathbb{E}[\mathsf{Err}(\mathcal{T})] \le \varepsilon$ *and*

$$D \le O((p^{7/6} \cdot w + p^{-11/3}) \cdot \log^2(w/\varepsilon))$$
$$M \le O(p^{-11/6}).$$

Proof. Let $\mathcal{X}^+$ be the set of (indices of) variables with fan-out more than $t \stackrel{\text{def}}{=} \lceil p^{-7/6} \rceil$. For each string $u \in \{0,1\}^{\mathcal{X}^+}$, let f_u be the restricted circuit obtained from f by replacing each variable x_i where $i \in \mathcal{X}^+$ with the constant u_i. Then f_u is a depth-d threshold circuit with at most w wires in which each variable has fan-out at most t, so we can apply Lemma 5.3.11 to it, giving a tree $\mathcal{T}_u$. Our tree $\mathcal{T}$ queries the variables in $\mathcal{X}^+ \cap Z$. Together with the restriction R, these queries determine a string $u \in \{0,1\}^{\mathcal{X}^+}$, and then $\mathcal{T}$ simulates $\mathcal{T}_u$.

Since f has at most w wires, $|\mathcal{X}^+| \le p^{7/6} \cdot w$. Therefore, the number of variable queries made by $\mathcal{T}$ is at most

$$O(p^{7/6} \cdot w \cdot \log^2(w/\varepsilon) + p^{-5/2} \cdot t \cdot \log(w/\varepsilon)) = O(p^{7/6} \cdot w \cdot \log^2(w/\varepsilon) + p^{-11/3} \cdot \log(w/\varepsilon)).$$

Meanwhile, the number of threshold function queries is at most $O(p^{-2/3} \cdot t) = O(p^{-11/6})$. Finally, let us bound $\mathbb{E}[\mathsf{Err}(\mathcal{T})]$. We have

$$\begin{aligned}
\mathbb{E}[\mathsf{Err}(\mathcal{T})] = \mathbb{E}\left[\mathsf{Err}(\mathcal{T}, U_Z)\right] &= \mathbb{E}\left[\mathsf{Err}(\mathcal{T}_{U_{\mathcal{X}^+}}, U_{Z \setminus \mathcal{X}^+})\right] \\
&= \mathbb{E}\left[\sum_{u \in \{0,1\}^{\mathcal{X}^+}} \mathsf{Err}(\mathcal{T}_u, U_{Z \setminus \mathcal{X}^+}) \cdot \mathbf{1}[U_{\mathcal{X}^+} = u]\right] \\
&= \sum_{u \in \{0,1\}^{\mathcal{X}^+}} \mathbb{E}\left[\mathsf{Err}(\mathcal{T}_u, U_{Z \setminus \mathcal{X}^+}) \cdot \mathbf{1}[U_{\mathcal{X}^+} = u]\right].
\end{aligned}$$

For a fixed u, the tree $\mathcal{T}_u$ ignores all the variables in $\mathcal{X}^+$. Therefore, the event $U_{\mathcal{X}^+} = u$ is

independent of the random variable $\mathsf{Err}(\mathcal{T}_u, U_{Z\setminus\mathcal{X}^+})$. Therefore,

$$\mathbb{E}[\mathsf{Err}(\mathcal{T})] = \sum_{u\in\{0,1\}^{\mathcal{X}^+}} \mathbb{E}\left[\mathsf{Err}(\mathcal{T}_u, U_{Z\setminus\mathcal{X}^+})\right] \cdot \Pr[U_{\mathcal{X}^+} = u] \leq 2^{|\mathcal{X}^+|} \cdot \varepsilon \cdot 2^{-|\mathcal{X}^+|} = \varepsilon. \qquad \square$$

5.3.3 Simplification of Threshold Circuits under Restrictions

So far, we have shown that hitting a depth-d threshold *circuit* with a suitable pseudorandom restriction yields a threshold decision tree where the circuits at the leaves have depth $d-1$. To continue on to depth $d-2$, $d-3$, etc., we need to analyze the effect of a pseudorandom restriction on a threshold *decision tree*. This is what we do in the next lemma.

Lemma 5.3.19 (Circuit-depth reduction for threshold decision trees)**.** *Let $f\colon \{\pm1\}^n \to \{\pm1\}$ be consistent with a (d, w, D, M, e)-threshold decision tree $\mathcal{T}$. Let $p, \varepsilon > 0$, let $k = p^{-5/6} \cdot (M + \log(12w/\varepsilon))$, and let $R = \mathsf{Res}(U, Z)$, where U is uniform random and Z is k-wise independent with marginals p. Then $f|_R$ is consistent with a $(d-1, w, D', M', e+w)$-threshold decision tree $\mathcal{T}'$, where*

$$\begin{aligned}
\mathbb{E}[\mathsf{Err}(\mathcal{T}')] &\leq \mathsf{Err}(\mathcal{T}) + \varepsilon \\
D' &\leq O(pD + (p^{7/6} \cdot w + p^{-11/3}) \cdot (M^2 + \log^2(w/\varepsilon))) \\
M' &\leq M + O(p^{-11/6}).
\end{aligned}$$

Proof. Recall that each leaf ℓ of $\mathcal{T}$ has a threshold circuit C_ℓ. By Lemma 5.3.18, $(C_\ell)_R$ is consistent with some $(d-1, w, D_0, M_0, w)$-threshold decision tree $\mathcal{T}_\ell$, where

$$\begin{aligned}
\mathbb{E}[\mathsf{Err}(\mathcal{T}_\ell)] &\leq \varepsilon \cdot 2^{-M-1} \\
D_0 &\leq O((p^{7/6} \cdot w + p^{-11/3}) \cdot (M^2 + \log^2(w/\varepsilon))) \\
M_0 &\leq O(p^{-11/6}).
\end{aligned}$$

Naturally, on input $u \in \{\pm 1\}^Z$, the tree $\mathcal{T}'$ begins by simulating the tree portion of $\mathcal{T}$, but with two changes. First, if $\mathcal{T}$ tries to query a variable outside Z, then $\mathcal{T}'$ just consults R to get the value instead of making a query. Second, we cap the number of variable queries at q for a certain value $q = O(pD + \log(1/\varepsilon))$. That is, if $\mathcal{T}'$ would need to make more than q variable queries for this simulation process, then $\mathcal{T}'$ instead halts after q variable queries; we reach a leaf ℓ' of $\mathcal{T}'$ that is labeled $C_{\ell'} \equiv 0$ and $\mathsf{Err}_{\ell'} \equiv 1$. Otherwise, the simulation reaches some leaf ℓ of $\mathcal{T}$, and from there $\mathcal{T}'$ simulates $\mathcal{T}_\ell$. Upon reaching a leaf ℓ' of $\mathcal{T}_\ell$ labeled with $C_{\ell'}$ and $\mathsf{Err}_{\ell'}$, the tree $\mathcal{T}'$ halts at its own leaf ℓ'' and sets $C_{\ell''} = C_{\ell'}$ and $\mathsf{Err}_{\ell''} = \mathsf{Err}_\ell + \mathsf{Err}_{\ell'}$.

Clearly, our tree $\mathcal{T}'$ is a $(d-1, w, D', M', e+w)$-threshold decision tree, where D' and M' are as in the lemma statement, and $f|_R$ is consistent with $\mathcal{T}'$. Now let us bound $\mathbb{E}[\mathsf{Err}(\mathcal{T}')] = \mathbb{E}_{U,Z}[\mathsf{Err}(\mathcal{T}', U_Z)]$. There are three sources of error in the computation of $\mathcal{T}'(U_Z)$. First, the tree might halt early to avoid making more than q variable queries while simulating the tree portion of $\mathcal{T}$. Whether this event occurs depends only on Z. By tail bounds for k-wise independence (Theorem 5.3.8), for a suitable choice of $q = O(pD + \log(1/\varepsilon))$, the probability of this bad event is at most $\varepsilon/2$, hence it contributes at most $\varepsilon/2$ to $\mathbb{E}[\mathsf{Err}(\mathcal{T}')]$. Second, there are the errors of $\mathcal{T}$ itself. This contributes at most $\mathsf{Err}(\mathcal{T})$ to $\mathbb{E}[\mathsf{Err}(\mathcal{T}')]$.

Third, most interestingly, there are the errors due to the tree $\mathcal{T}_\ell$. These errors are less straightforward to bound, because the event of reaching some particular leaf ℓ of $\mathcal{T}$ might be *correlated* with the unlikely event that an error occurs in the computation of $\mathcal{T}_\ell$. The main reason for these potential correlations is the threshold function queries made by $\mathcal{T}$. Because of those queries, every single bit of the assignment U might play a role in determining the leaf that we reach, including the bits that $\mathcal{T}_\ell$ reads.

To bound the harmful effect of these correlations, let L be the set of leaves of $\mathcal{T}$, and

let $\ell(u)$ denote the leaf that $\mathcal{T}$ reaches on input u. Our goal is to bound

$$\mathop{\mathbb{E}}_{U,Z}[\mathsf{Err}(\mathcal{T}_{\ell(U)}, U_Z)] = \sum_{\ell \in L} \mathop{\mathbb{E}}_{U,Z}[\mathsf{Err}(\mathcal{T}_\ell, U_Z) \cdot \mathbf{1}[\ell(U) = \ell]].$$

For a leaf $\ell \in L$ and an input $u \in \{\pm 1\}^n$, let us say that u is *variable-query-consistent* with ℓ if for each vertex on the path from the root to ℓ that makes a variable query – say the vertex queries x_i and the outgoing edge labeled b is on the path to ℓ – we have $u_i = b$. (Note that this definition ignores the threshold function queries made by $\mathcal{T}$.) Obviously each input u is variable-query-consistent with $\ell(u)$, so

$$\mathop{\mathbb{E}}_{U,Z}[\mathsf{Err}(\mathcal{T}_{\ell(U)}, U_Z)] \leq \sum_{\ell \in L} \mathop{\mathbb{E}}_{U,Z}[\mathsf{Err}(\mathcal{T}_\ell, U_Z) \cdot \mathbf{1}[U \text{ is variable-query-consistent with } \ell].$$

Now, for a fixed leaf ℓ, without loss of generality, the tree $\mathcal{T}_\ell$ ignores every variable that is queried on the path from the root to ℓ. Therefore, the event that U is variable-query-consistent with ℓ is independent of the random variable $\mathsf{Err}(\mathcal{T}_\ell, U_Z)$, so

$$\begin{aligned}
\mathop{\mathbb{E}}_{U,Z}[\mathsf{Err}(\mathcal{T}_{\ell(U)}, U_Z)] &\leq \sum_{\ell \in L} \mathop{\mathbb{E}}_{U,Z}[\mathsf{Err}(\mathcal{T}_\ell, U_Z)] \cdot \Pr_U[U \text{ is variable-query-consistent with } \ell] \\
&= \sum_{\ell \in L} \mathop{\mathbb{E}}_{U,Z}[\mathsf{Err}(\mathcal{T}_\ell)] \cdot \Pr_U[U \text{ is variable-query-consistent with } \ell] \\
&\leq \varepsilon \cdot 2^{-M-1} \cdot \sum_{\ell \in L} \Pr_U[U \text{ is variable-query-consistent with } \ell] \\
&= \varepsilon \cdot 2^{-M-1} \cdot \mathop{\mathbb{E}}_U\left[|\{\ell \in L : U \text{ is variable-query-consistent with } \ell\}|\right].
\end{aligned}$$

For a fixed $u \in \{\pm 1\}^n$, let us bound the number of leaves ℓ with which u is variable-query-consistent. To find such a leaf ℓ, we must start at the root. Each time we reach a node that makes a variable query, say to x_i, we must follow the outgoing edge labeled u_i. On the other hand, when we reach a node that makes a threshold function query, we can freely choose

either outgoing edge. We make up to M binary choices in this way, so there are at most 2^M leaves we might reach, i.e., there are at most 2^M leaves with which u is variable-query-consistent. Therefore,

$$\mathop{\mathbb{E}}_{U,Z}[\mathsf{Err}(\mathcal{T}_{\ell(U)}, U_Z)] \leq \varepsilon \cdot 2^{-M-1} \cdot 2^M \leq \varepsilon/2.$$

Summing up, this shows that

$$\mathbb{E}[\mathsf{Err}(\mathcal{T}')] \leq \varepsilon/2 + \mathsf{Err}(\mathcal{T}) + \varepsilon/2 = \mathsf{Err}(\mathcal{T}) + \varepsilon$$

as claimed. □

Iterating the circuit-depth-reduction lemma

The last lemma shows that under suitable pseudorandom restriction, a threshold decision tree with circuit-depth d becomes a threshold decision tree with circuit-depth $d-1$. Iterating this procedure $d-1$ times, we obtain a threshold decision tree with circuit-depth 1, i.e., the "circuits" at the leaves are simply individual threshold functions. The following lemma records the parameters of the tree we obtain in this manner, starting (for simplicity and because it suffices for our purposes) from a threshold circuit rather than a threshold decision tree. This is our final simplification lemma for threshold circuits.

Lemma 5.3.20 (Overall simplification of threshold circuits)**.** *Let $w, d \in \mathbb{N}$ and $p, \varepsilon > 0$. There is a distribution over subsets $Z \subseteq [n]$ that can be sampled explicitly using*

$$O(p^{-1} \cdot d \cdot \log(w/\varepsilon) \cdot \log n)$$

truly random bits such that $|Z| = \lceil pn \rceil$, and if we sample U uniformly at random and let

$R = \mathsf{Res}(U, Z)$, then for any depth-d threshold circuit f with w wires, the restricted function $f|_R$ is consistent with a $(1, n, D, M, dw)$-threshold decision tree $\mathcal{T}$, where $\mathbb{E}[\mathsf{Err}(\mathcal{T})] \leq \varepsilon$ and

$$D \leq (p^{1+40^{-(d-1)}} \cdot w + p^{-4}) \cdot 2^{O(d)} \cdot \log^2(w/\varepsilon)$$
$$M \leq O(p^{-2} \cdot d).$$

Lemma 5.3.20 can be viewed as a derandomized and strengthened form of some of the analysis by Chen, Santhanam, and Srinivasan [CSS18]. The proof of Lemma 5.3.20 mainly consists of picking parameters and is not conceptually interesting. We will therefore omit the proof; the interested reader can read the proof in our paper [HHTT21, Proposition 6.11]. In lieu of a full proof, we will briefly discuss the most interesting part of the calculation: in the first term of the bound on D, where does the exponent $1 + 2^{-\Theta(d)}$ come from? A larger exponent would translate to a PRG with a better seed length.

As we said earlier, the proof of Lemma 5.3.20 works by applying Lemma 5.3.19 $d-2$ times to decrease the circuit-depth all the way from d down to 1. We use a different value of p in each round; let p_i be the $\star$-probability that we use when going from circuit-depth i to circuit-depth $i-1$. Looking at the bounds in Lemma 5.3.19, we see that for that restriction, the number of threshold function queries goes from M_i to $M_{i-1} \approx M_i + p^{-2}$, and the number of variable queries goes from D_i to D_{i-1}, where

$$D_{i-1} \approx p_i \cdot D_i + p_i^{7/6} \cdot w \cdot (M_i^2 + 1). \tag{5.12}$$

In the first round, the initial "tree" is actually just a threshold circuit, so $D_d = M_d = 0$, hence $M_{d-1} \approx p_d^{-2}$ and $D_{d-1} \approx p_d^{7/6} \cdot w$. In the second round, things get a little more

complicated. Focusing only on the second term of Eq. (5.12), we get

$$D_{d-2} > p_{d-1}^{7/6} \cdot w \cdot M_{d-1}^2 \approx p_{d-1}^{7/6} \cdot p_d^{-2} \cdot w.$$

To get a nontrivial result, we are forced to choose p_{d-1} much smaller than p_d, so that the $+7/6$ power drowns out the -2 power. In particular, we should choose $p_{d-1} < p_d^C$ for a sufficiently large constant $C > 1$. (In the actual proof, we set $p_{d-1} \approx p_d^{40}$ [HHTT21].) Something similar happens in each subsequent round, forcing us to choose the $\star$-probabilities to be rapidly decreasing from one round to the next, with $p_{i-1} < p_i^C$. When we compose all these restrictions, the overall $\star$-probability is the product $p_d \cdot p_{d-1} \cdots p_2$. To achieve overall $\star$-probability p, therefore, we must start with p_d much larger than p, namely $p_d = p^{2^{-\Theta(d)}}$. Looking now at the first term of Eq. (5.12), we see that $D_2 \geq p_2 \cdot p_3 \cdots p_{d-1} \cdot p_d^{7/6} \cdot w = p^{1+2^{-\Theta(d)}} \cdot w$, which explains why we are not able to beat exponent $1 + 2^{-\Theta(d)}$.

5.3.4 PRGs for $\mathsf{ANY}_m \circ \mathsf{THR}$ and the Simplified Model

So far, we have shown that under pseudorandom restrictions threshold circuits simplify in some sense. Ironically, our notion of "simplicity," based on the threshold decision tree model, is rather *complicated*. But just like when we were studying read-once $\mathbf{AC}^0$, what really matters is whether we can design a PRG for the simplified model. Our goal for this section is to show that we can.

Fooling $\mathsf{ANY}_m \circ \mathsf{THR}$ with extremely low error

We begin by studying PRGs for *functions of a few threshold functions*. Let $\mathsf{ANY}_m \circ \mathsf{THR}$ be the class of functions f of the form $f(x) = g(\Phi_1(x), \ldots, \Phi_m(x))$, where $g \colon \{\pm 1\}^m \to \{\pm 1\}$ is an arbitrary function and $\Phi_1, \ldots, \Phi_m \colon \{\pm 1\}^n \to \{\pm 1\}$ are threshold functions. Let $\mathsf{AND}_m \circ \mathsf{THR}$ be the subclass where $g = \mathsf{AND}$ (note that the ± 1 version of the AND function

is the max function). The latter functions are also known as *polytopes*. Ultimately we will only need a PRG for $\mathsf{AND}_m \circ \mathsf{THR}$, but we will present an explicit PRG for the more powerful class $\mathsf{ANY}_m \circ \mathsf{THR}$ with seed length

$$\widetilde{O}\left(\sqrt{n \cdot (m + \log(1/\varepsilon))}\right).$$

For moderate values of m and ε, our seed length is far from the optimal seed length of $O(m + \log(n/\varepsilon))$. However, note that our seed length remains *nontrivial* provided

$$m + \log(1/\varepsilon) < n/\log^c n$$

for some constant c, and in this respect our PRG is near-optimal, because nontrivial PRGs do not exist when $m \geq n$ or $\log(1/\varepsilon) \geq n$. We will be interested in the regime $m \approx n^{0.2}$ and $\varepsilon = 2^{-n^{1-\alpha}}$ for an arbitrarily small constant α; for these parameters, our seed length is indeed slightly nontrivial. Several previous works [Vio07; GOWZ10; LS11; HKM12; ST17b; ST18; OST19; CDS19; KKLMO20] have studied PRGs for functions of a few threshold functions and related problems, but none of them give nontrivial PRGs in the extreme parameter regime in which we are interested (see Table 5.3).

Our PRG uses some similar techniques as the earlier PRG by Kabanets, Koroth, Lu, Myrisiotis, and Oliveira [KKLMO20]. Like them, the only fact we use about threshold functions is that they have low *communication complexity*. Specifically, we consider the multiparty "number-in-hand" model, i.e., the input is split into several disjoint pieces, each piece is held by a single party, and when one party speaks, all the parties can hear what they say; the identity of the speaker in a given round is a function of the communication in the prior rounds. Nisan designed efficient protocols for threshold functions in this model [Nis93b], and later Viola gave an improvement [Vio15].

Seed length	Model fooled	Reference
$\widetilde{O}((m+\log(n/\varepsilon))\cdot\log(1/\varepsilon))$	$\mathsf{MONOTONE}_m \circ \mathsf{THR}$	[GOWZ10]
$O(m\cdot\log(n/\varepsilon)\cdot\log n)$	$\mathsf{MONOTONE}_m \circ \mathsf{THR}$	[GOWZ10]
$m^{O(1)}+\log^{O(1)}(1/\varepsilon)+2^{O(\sqrt{\log n})}$	$\mathbf{AC}^0$ with m THR gates	[Vio07; LS11; ST18]
$\mathrm{poly}(\log m, 1/\varepsilon)\cdot\log n$	$\mathsf{AND}_m \circ \mathsf{THR}$	[OST19]
$\widetilde{O}(\sqrt{n}\cdot m^{1/4}\cdot\log(1/\varepsilon))$	$\mathsf{FORMULA}[m] \circ \mathsf{THR}$	[KKLMO20]
$\widetilde{O}\left(\sqrt{n\cdot(m+\log(1/\varepsilon))}\right)$	$\mathsf{ANY}_m \circ \mathsf{THR}$	[HHTT21] (Theorem 5.3.24)

Table 5.3: Explicit PRGs for functions of a few unrestricted threshold functions. $\mathsf{MONOTONE}_m$ denotes the class of monotone Boolean functions on m input bits, and $\mathsf{FORMULA}[m]$ denotes the class of De Morgan formulas with m leaves. Note that an ε-PRG for $\mathsf{AND}_m \circ \mathsf{THR}$ is automatically an $(\varepsilon \cdot 2^m)$-PRG for $\mathsf{ANY}_m \circ \mathsf{THR}$, so each of these PRGs implies *some* nontrivial PRG for $\mathsf{ANY} \circ \mathsf{THR}$. However, even for the special case of $\mathsf{AND}_m \circ \mathsf{THR}$, none of the prior PRGs are nontrivial for the regime of parameters in which we are interested, namely $m \approx n^{0.2}$ and $\varepsilon = 2^{-n^{1-\alpha}}$, because each prior seed length is at least $\min\{m, \log(1/\varepsilon)\}\cdot\log(1/\varepsilon)$.

Theorem 5.3.21 ([Nis93b; Vio15]). *For any threshold function $\Phi\colon \{\pm 1\}^n \to \{\pm 1\}$, for any $\delta > 0$, under any partition of the input bits into k parts, there is a public-coin randomized k-party number-in-hand protocol for Φ with failure probability δ and communication complexity $O(k \cdot \log k \cdot \log(n/\delta))$.*

We can fool a multiparty number-in-hand communication protocol using a PRG for *combinatorial rectangles*. Recall that a k-dimensional combinatorial rectangle is a function $f\colon (\{\pm 1\}^{n/k})^k \to \{0,1\}$ of the form $f(x^{(1)}, \ldots, x^{(k)}) = f_1(x^{(1)}) \wedge \cdots \wedge f_k(x^{(k)})$, where each f_i is an arbitrary function $f_i\colon \{\pm 1\}^{n/k} \to \{0,1\}$. There is a long line of work developing PRGs for combinatorial rectangles [ASWZ96; LLSZ97; EGLNV98; Lu02; GMRTV12; Vio14; GY20]. The best seed length is by Gopalan and Yehudayoff [GY20], which translates to the following parameters for fooling multiparty communication protocols.

Theorem 5.3.22 ([GY20]). *Let $n, k, t \in \mathbb{N}$ and $\varepsilon > 0$ with n a multiple of k. There is an explicit ε-PRG for the class of functions $f\colon (\{\pm 1\}^{n/k})^k \to \{\pm 1\}$ such that $f(x^{(1)}, \ldots, x^{(k)})$ can be computed by a deterministic k-party number-in-hand protocol where party i holds $x^{(i)}$ with communication complexity t. The seed length of the PRG is*

$$\widetilde{O}(t + n/k + \log(1/\varepsilon) + \log\log k).$$

Proof. Gopalan and Yehudayoff show how to ε'-fool k-dimensional combinatorial rectangles with a seed length of $\widetilde{O}(n/k + \log(1/\varepsilon') + \log\log k)$ [GY20]. Let A be the set of transcripts $z \in \{0,1\}^t$ that cause the protocol to output -1. For each transcript z, there is some combinatorial rectangle f_z such that $f_z(x^{(1)}, \ldots, x^{(k)}) = 1$ if and only if z is the transcript of the protocol on input $(x^{(1)}, \ldots, x^{(k)})$. Therefore,

$$f(x) = (-1)^{\bigvee_{z \in A} f_z(x)} = 1 - 2 \cdot \sum_{z \in A} f_z(x).$$

It follows that any ε'-PRG for k-dimensional combinatorial rectangles fools f with error $2\varepsilon' \cdot |A| \leq \varepsilon' \cdot 2^{t+1}$. Setting $\varepsilon' = \varepsilon \cdot 2^{-t-1}$ completes the proof. □

At this point, to get a PRG for $\mathsf{ANY}_m \circ \mathsf{THR}$, we could try reasoning that since an individual threshold function has low communication complexity, a function in $\mathsf{ANY}_m \circ \mathsf{THR}$ must also have low communication complexity, and so we can fool it using the PRG for communication protocols. The trouble with this argument is that the protocol for an individual threshold function is *randomized*. To get a PRG with error ε, naïvely we need to use a protocol with failure probability less than ε, which naïvely would translate to a PRG seed length of more than $m \cdot \log(1/\varepsilon)$. That would be too large for us.

To get a better seed length, we will use a special polynomial representation of the outer ANY_m function, which was also the approach of like Kabanets, Koroth, Lu, Myrisiotis, and Oliveira [KKLMO20]. Kabanets et al. used a low-degree approximating polynomial, whereas we use a beautiful theorem by Sherstov on polynomials that are *robust to noise* at the inputs [She13]. For a multivariate real polynomial $p(x) = \sum_e c_e \cdot x_1^{e_1} \cdots x_n^{e_n}$, we define $\deg(p) = \max\{e_1 + \cdots + e_n : c_e \neq 0\}$ and $L_1(p) = \sum_e |c_e|$.

Theorem 5.3.23 ([She13])**.** *For any multivariate real polynomial $p\colon \{\pm 1\}^n \to [-1, 1]$ and any $\delta > 0$, there is a multivariate real polynomial $p_{\text{robust}}\colon \mathbb{R}^n \to \mathbb{R}$ such that for every $x \in \{\pm 1\}^n$ and every $E \in [-1/3, 1/3]^n$,*

$$|p(x + E) - f(x)| \leq \delta.$$

Furthermore, $\deg(p_{\text{robust}}) \leq O(\deg(p) + \log(1/\delta))$ *and*[7] $L_1(p_{\text{robust}}) \leq L_1(p) \cdot 2^{O(\deg(p)+\log(1/\delta))}$.

[7] Sherstov did not explicitly bound $L_1(p_{\text{robust}})$, but the bound can be verified by looking at his construction [She13], as we now explain. In the proof of Sherstov's Theorem 5.2, the polynomial he denotes p and the values he denotes as d and D satisfy $L_1(p) \leq 2^{O(d+D)}$. Therefore, in the proof of his Theorem 6.2, we get $L_1(P) \leq L_1(\phi) \cdot 2^{O(D+\deg(\phi))} = L_1(\phi) \cdot 2^{O(\deg(\phi)+\log(1/\delta))}$. Composing with the polynomial r at the end of that proof only increases $L_1(p_{\text{robust}})$ by a constant factor.

Sherstov's result will allow us to use communication protocols for threshold functions with a *constant* failure probability, $\delta = 1/6$, while nevertheless achieving a low PRG error ε. In particular, we now show that PRGs for combinatorial rectangles, with suitable parameters, fool $\mathsf{ANY}_m \circ \mathsf{THR}$ with seed length $\widetilde{O}(\sqrt{n \cdot (m + \log(1/\varepsilon))})$.

Theorem 5.3.24. *For any $n, m \in \mathbb{N}$ and any $\varepsilon > 0$, there is an explicit ε-PRG for $\mathsf{ANY}_m \circ \mathsf{THR}$ with seed length $\widetilde{O}\left(\sqrt{n \cdot (m + \log(1/\varepsilon))}\right)$.*

Proof. The construction is the γ-PRG for k-party number-in-hand protocols with communication complexity t of Theorem 5.3.22, where γ, k, and t will be chosen later. Consider any function of the form $f(x) = p(\Phi_1(x), \ldots, \Phi_m(x))$, where $\Phi_1, \ldots, \Phi_m$ are threshold functions and $p\colon \{\pm 1\}^n \to \{\pm 1\}$ is an arbitrary function. For each $i \in [m]$, let π_i be the randomized communication protocol for Φ_i with failure probability $1/6$ guaranteed by Theorem 5.3.21, with communication complexity $t_0 = O(k \cdot \log k \cdot \log n)$. Write the output of π_i as $\pi_i(x, Y)$, where x is the input and Y is the random bits. Define $q_i(x) = \mathbb{E}_Y[\pi_i(x, Y)] \in [-1, 1]$, i.e., $q_i(x)$ is the expected output of π_i on input x, and hence $|q_i(x) - \Phi_i(x)| \leq 1/3$.

Identify p with its Fourier expansion, a multilinear polynomial. Let p_{robust} be the corresponding polynomial from Theorem 5.3.23 with error value $\delta = \varepsilon/2$. That way, for every $x \in \{\pm 1\}^n$,

$$|f(x) - p_{\text{robust}}(q_1(x), \ldots, q_m(x))| \leq \varepsilon/2.$$

It suffices, therefore, to fool the function $p_{\text{robust}}(q_1(x), \ldots, q_m(x))$. Expanding p_{robust}, we can write

$$p_{\text{robust}}(q_1(x), \ldots, q_m(x)) = \sum_{e \in \mathbb{Z}^m_{\geq 0}} c_e \cdot q_1(x)^{e_1} \cdots q_m(x)^{e_m}.$$

Consider a fixed exponent vector e with $c_e \neq 0$. There is a probabilistic interpretation of the quantity $q_1(x)^{e_1} \cdots q_m(x)^{e_m}$. That quantity is precisely the expected value of a new randomized protocol Π_e that simulates each q_i protocol e_i times independently and takes the

product (XOR) of all the output values. This new protocol has communication complexity

$$\begin{aligned} t_0 \cdot \deg(p_{\text{robust}}) &\leq O(k \cdot \log k \cdot \log n \cdot (\deg(p) + \log(1/\varepsilon)) \\ &\leq O(k \cdot \log k \cdot \log n \cdot (m + \log(1/\varepsilon))). \end{aligned}$$

Therefore, choosing $t = \widetilde{O}(k \cdot (m + \log(1/\varepsilon)) \cdot \log n)$, our pseudorandom distribution X satisfies

$$\begin{aligned} \left| \mathbb{E}\left[\prod_{i=1}^{m} q_i(X)^{e_i}\right] - \mathbb{E}\left[\prod_{i=1}^{m} q_i(U_n)^{e_i}\right] \right| &= \left| \mathop{\mathbb{E}}_{X,Y}[\Pi_e(X,Y)] - \mathop{\mathbb{E}}_{U_n,Y}[\Pi_e(U_n,Y)] \right| \\ &\leq \mathop{\mathbb{E}}_{Y}\left[\left| \mathop{\mathbb{E}}_{X}[\Pi_e(X,Y)] - \mathop{\mathbb{E}}_{U_n}[\Pi_e(U_n,Y)] \right| \right] \\ &\leq \gamma. \end{aligned}$$

As a consequence, summing up all the monomials of p_{robust}, we get

$$\begin{aligned} |\mathbb{E}[p_{\text{robust}}(q_1(X), \ldots, q_m(X))] - \mathbb{E}[p_{\text{robust}}(q_1(U_n), \ldots, q_m(U_n))]| &\leq \gamma \cdot L_1(p_{\text{robust}}) \\ &\leq \gamma \cdot L_1(p) \cdot 2^{O(\deg(p)+\log(1/\varepsilon))} \\ &\leq \gamma \cdot L_1(p) \cdot 2^{O(m+\log(1/\varepsilon))}. \end{aligned}$$

By Parseval's theorem (see, e.g., O'Donnell's text [O'D14]) and the Cauchy-Schwarz inequality, we have $L_1(p) \leq 2^{m/2}$. Therefore, X fools $p_{\text{robust}}(q_1(X), \ldots, q_m(X))$ with error $\gamma \cdot 2^{O(m+\log(1/\varepsilon))}$, so a suitable $\gamma = \varepsilon \cdot 2^{-O(m+\log(1/\varepsilon))}$ ensures that X fools f with error ε. With this choice of t and γ, the seed length of the PRG is

$$\widetilde{O}(t + n/k + \log(1/\gamma) + \log\log k) = \widetilde{O}(n/k + k \cdot (m + \log(1/\varepsilon)) \cdot \log n).$$

To balance the two terms, we pick $k = \sqrt{n/(m + \log(1/\varepsilon))}$, giving the desired seed length.

□

Fooling threshold decision trees

Our PRG for $\mathsf{ANY}_m \circ \mathsf{THR}$ readily implies a PRG for threshold decision trees when the leaf-circuits are individual threshold functions, as we now explain.

Lemma 5.3.25. *For any $n, D, M, e \in \mathbb{N}$ and any $\varepsilon > 0$, there is an explicit PRG G such that if $f\colon \{\pm 1\}^n \to \{\pm 1\}$ is consistent with some $(1, n, D, M, e)$-threshold tree $\mathcal{T}$, then G fools f with error $\varepsilon + 4 \cdot \mathsf{Err}(\mathcal{T})$. Furthermore, G has seed length*

$$\widetilde{O}\left(\sqrt{n \cdot (D + M + \log(e+1) + \log(1/\varepsilon))}\right).$$

Proof. Let G be a γ-PRG for $\mathsf{AND}_{M+2} \circ \mathsf{THR}$, where γ will be chosen later. For $b \in \{\pm 1\}$, define $f_b(x) = \mathcal{T}(x) + 2b \cdot \mathsf{Err}(\mathcal{T}, x)$. Let L be the set of leaves of $\mathcal{T}$, and let $\ell(x)$ be the leaf reached on input x. Recall that each leaf ℓ is labeled with a circuit $C_\ell(x)$ and an error function $\mathsf{Err}_\ell(x)$. The error function is of the form $\mathsf{Err}_\ell(x) = \sum_{i=1}^{e} \mathbf{1}[\Phi_{\ell,i}(x) = -1]$, where $\Phi_{\ell,i}$ is a threshold function. The "circuit" C_ℓ is actually just a single threshold function. Thus, we have

$$\begin{aligned}
f_b(x) &= \sum_{\ell \in L} (C_\ell(x) + 2b \cdot \mathsf{Err}_\ell(x)) \cdot \mathbf{1}[\ell(x) = \ell] \\
&= \sum_{\ell \in L} \left(1 - 2 \cdot \mathbf{1}[C_\ell(x) = -1] + 2b \cdot \sum_{i=1}^{e} \mathbf{1}[\Phi_{\ell,i}(x) = -1]\right) \cdot \mathbf{1}[\ell(x) = \ell] \\
&= |L| - 2 \cdot \sum_{\ell \in L} \mathbf{1}[C_\ell(x) = -1] \cdot \mathbf{1}[\ell(x) = \ell] + 2b \cdot \sum_{\ell \in L} \sum_{i=1}^{e} \mathbf{1}[\Phi_{\ell,i}(x) = -1] \cdot \mathbf{1}[\ell(x) = \ell].
\end{aligned}$$

For a fixed leaf ℓ, the function $\mathbf{1}[\ell(x) = \ell]$ is in $\mathsf{AND}_{M+1} \circ \mathsf{THR}$, since all the variable queries on the path to ℓ can be joined into a single threshold function (a conjunction of literals).

Therefore, each summand is in $\mathsf{AND}_{M+2} \circ \mathsf{THR}$, so G fools f_b with error $2|L| \cdot \gamma + 2|L| \cdot e \cdot \gamma \leq 2^{D+M+1} \cdot (e+1) \cdot \gamma$. Choose $\gamma = \varepsilon \cdot 2^{-(D+M+1)}/(e+1)$, so G fools f_b with error ε.

It follows that G fools f, because f is sandwiched between f_{-1} and f_{+1}. In more detail, if we let $X = G(U_s)$ where s is the seed length of G, then

$$\begin{aligned}
\mathbb{E}[f(X)] \leq \mathbb{E}[\mathcal{T}(X) + 2\mathsf{Err}(\mathcal{T}, X)] &= \mathbb{E}[f_{+1}(X)] \\
&\leq \mathbb{E}[f_{+1}(U_n)] + \varepsilon \\
&= \mathbb{E}[\mathcal{T}(U_n) + 2 \cdot \mathsf{Err}(\mathcal{T}, U_n)] + \varepsilon \\
&= \mathbb{E}[\mathcal{T}(U_n) - 2 \cdot \mathsf{Err}(\mathcal{T}, U_n)] + \varepsilon + 4 \cdot \mathsf{Err}(\mathcal{T}) \\
&\leq \mathbb{E}[f(U_n)] + \varepsilon + 4 \cdot \mathsf{Err}(\mathcal{T}),
\end{aligned}$$

where the first and last steps both used the fact that f is consistent with $\mathcal{T}$. A perfectly analogous argument shows that $\mathbb{E}[f(X)] \geq \mathbb{E}[f(U_n)] - \varepsilon - 4 \cdot \mathsf{Err}(\mathcal{T})$.

Taking G to be the PRG for $\mathsf{ANY}_{M+2} \circ \mathsf{THR}$ of Theorem 5.3.24, the seed length is

$$\widetilde{O}\left(\sqrt{n \cdot (M + 2 + \log(1/\gamma))}\right) = \widetilde{O}\left(\sqrt{n \cdot (D + M + \log(e+1) + \log(1/\varepsilon))}\right)$$

as claimed. □

5.3.5 The Final PRG via the Ajtai-Wigderson Framework

Construction

At this point, we have shown that for a truly random U and a pseudorandom Z, the restriction $\mathsf{Res}(U, Z)$ simplifies threshold circuits to a model we can fool with a nontrivial seed length. Sampling U would be too expensive, but we can nevertheless get a PRG from here using the method of Ajtai and Wigderson [AW89]. The idea is, we sample Z, and we assign

values to the coordinates in Z using the PRG for the simplified model, but then for the coordinates outside Z, instead of sampling a truly random string U, we recursively sample a pseudorandom string that fools threshold circuits. The number of pseudorandom bits we need has decreased from n to $(1-p)\cdot n$, so there are roughly p^{-1} levels of recursion, which is why we needed our failure probability to be less than p in each iteration.

In more detail, fix $w, d \in \mathbb{N}$ and $\gamma, p \in (0, 1/2)$. For each $n \in \mathbb{N}$, we will recursively define a distribution $X^{(n)}$ over $\{\pm 1\}^n$ that fools depth-d threshold circuits with w wires with error γ_n, where the values γ_n will become clear later. When $n = 1$, we set $X^{(n)} = U_1$. Now consider $n > 1$. Let $n' = \lceil p \cdot n \rceil$. Sample $Z \subseteq [n]$ according to the distribution of Lemma 5.3.20 with $\varepsilon = \gamma$, so $|Z| = \lceil pn \rceil$. Sample $Y \in \{\pm 1\}^Z$ using the PRG of Lemma 5.3.25, where the parameters D and M are as in the conclusion of Lemma 5.3.20 and $e = dw$ and $\varepsilon = \gamma$. For convenience, extend Y to a string $Y' \in \{\pm 1\}^n$ by putting $+1$ in the coordinates outside Z. Meanwhile, independently, sample $X \sim X^{(n-|Z|)} \in \{\pm 1\}^{n-|Z|}$ recursively. Define $X' \in \{\pm 1\}^n$ so that $X'_{[n]\setminus Z} = X$ and $X_Z = (+1)^{|Z|}$. Finally, we set

$$X^{(n)} = \mathsf{Res}(X', Z) \circ Y' = \mathsf{Res}(Y', [n] \setminus Z) \circ X'.$$

Correctness

Claim 5.3.26. *For each $n \in \mathbb{N}$, $X^{(n)}$ fools depth-d threshold circuits with at most w wires with error γ_n, where $\gamma_n = O(p^{-1} \cdot \log n \cdot \gamma)$.*

Proof. Let f be a depth-d threshold circuit with at most w wires. This class is closed under restriction, so $f|_{\mathsf{Res}(Y',[n]\setminus Z)}$ is also a depth-d threshold circuit with at most w wires. The distribution $X^{(n-|Z|)}$ fools such circuits, so if we sample $U \in \{\pm 1\}^n$ uniformly at random,

then

$$\begin{aligned}\left|\mathbb{E}[f(X^{(n)})] - \mathbb{E}[f(\mathsf{Res}(U,Z)\circ Y')]\right| &= \left|\mathbb{E}[f|_{\mathsf{Res}(Y',[n]\setminus Z)}(X')] - \mathbb{E}[f|_{\mathsf{Res}(Y',[n]\setminus Z)}(U)]\right| \\ &\le \mathop{\mathbb{E}}_{Y',Z}\left[\left|\mathop{\mathbb{E}}_{X'}[f|_{\mathsf{Res}(Y',[n]\setminus Z)}(X')] - \mathop{\mathbb{E}}_{U}[f|_{\mathsf{Res}(Y',[n]\setminus Z)}(U)]\right|\right] \\ &\le \gamma_{n-|Z|}.\end{aligned}$$

Meanwhile, by Lemma 5.3.20, when we hit f with the restriction $\mathsf{Res}(U,Z)$, the restricted function is consistent with some threshold decision tree $\mathcal{T}$ (a random variable that is independent of Y'). Therefore, if we sample a fresh uniform random string $U' \in \{\pm 1\}^n$, then

$$\begin{aligned}\left|\mathbb{E}[f(\mathsf{Res}(U,Z)\circ Y')] - \mathbb{E}[f(\mathsf{Res}(U,Z)\circ U')]\right| &= \left|\mathbb{E}[f|_{\mathsf{Res}(U,Z)}(Y')] - \mathbb{E}[f|_{\mathsf{Res}(U,Z)}(U')]\right| \\ &\le \mathop{\mathbb{E}}_{U,Z}\left[\left|\mathop{\mathbb{E}}_{Y'}[f|_{\mathsf{Res}(U,Z)}(Y')] - \mathop{\mathbb{E}}_{U'}[f|_{\mathsf{Res}(U,Z)}(U')]\right|\right] \\ &\le \mathop{\mathbb{E}}_{U,Z}[\gamma + 4\cdot \mathsf{Err}(\mathcal{T})] \\ &\le 5\gamma,\end{aligned}$$

where the second to last step is by Lemma 5.3.25. Now, $\mathsf{Res}(U,Z)\circ U')$ is simply a uniform random n-bit string, so overall, we get

$$\left|\mathbb{E}[f(X^{(n)})] - \mathbb{E}[f]\right| \le \gamma_{n-|Z|} + 5\gamma,$$

i.e., $X^{(n)}$ fools f with error γ_n where $\gamma_1 = 0$ and for $n > 1$, $\gamma_n \le \gamma_{n-\lceil pn\rceil} + 5\gamma$. It follows immediately that $\gamma_n \le O(p^{-1}\cdot \log n \cdot \gamma)$. □

Seed length and the $\star$-probability p

Set $p = w^{-1/10}$. By Lemmas 5.3.20 and 5.3.25, the seed length for sampling Y is given by

$$\widetilde{O}\left(\sqrt{pn \cdot (D + M + \log(wd) + \log(1/\gamma))}\right),$$

where

$$\begin{aligned} D &\leq (p^{1+40^{-(d-1)}} \cdot w + p^{-4}) \cdot 2^{O(d)} \cdot \log^2(w/\gamma) \\ &= p^{1+40^{-(d-1)}} \cdot w \cdot 2^{O(d)} \cdot \log^2(w/\gamma) \end{aligned}$$

and $M \leq O(p^{-2} \cdot d) \ll D$. Recalling the statement of Theorem 5.3.2, we will have $n \leq w$. Thus, the seed length for sampling Y is at most

$$\widetilde{O}\left(\sqrt{pw \cdot p^{1+40^{-(d-1)}} \cdot w \cdot 2^{O(d)} \cdot \log^2(w/\gamma)}\right) = \widetilde{O}\left(p^{1+\frac{1}{2}40^{-(d-1)}} \cdot w \cdot 2^{O(d)} \cdot \log(1/\gamma)\right).$$

The seed length for sampling Z is only $O(p^{-1} \cdot d \cdot \log(w/\gamma) \cdot \log n)$, which is much smaller, so the overall seed length for sampling $X^{(n)}$ is asymptotically the seed length for Y times the recursion depth of $O(p^{-1} \log n)$, giving

$$\widetilde{O}\left(p^{\frac{1}{2} \cdot 40^{-(d-1)}} \cdot w \cdot 2^{O(d)} \cdot \log(1/\gamma)\right).$$

We should set $\gamma = \varepsilon / O(p^{-1} \log n)$ where ε is the desired error of the PRG, so the seed length is bounded by

$$\widetilde{O}\left(w^{1-2 \cdot 40^{-d}} \cdot 2^{O(d)} \cdot \log(1/\varepsilon)\right).$$

Now, let $\delta = \frac{1}{2} \cdot 50^{-d}$. Recalling the statement of Theorem 5.3.2, we are interested in the case $w = n^{1+\delta}$ and $\varepsilon = 2^{-n^\delta}$, and we want to achieve seed length $O(n^{1-\delta})$. We can assume

without loss of generality that $\delta \geq 1/\log n$, since otherwise the desired seed length is bigger than n. Thus, $50^d \leq \frac{1}{2}\log n$, so $2^{O(d)} = \operatorname{polylog} n$, and so our PRG's seed length is at most

$$\widetilde{O}(w^{1-2\cdot 40^{-d}} \cdot \log(1/\varepsilon)) \leq n^{1+2\delta-2\cdot 40^{-d}} \cdot \operatorname{polylog} n.$$

Now, $1.5 \cdot 40^{-d} > 1.5 \cdot 50^{-d} = 3\delta$, so the seed length is at most

$$n^{1-\delta} \cdot n^{-0.5\cdot 40^{-d}} \cdot \operatorname{polylog} n.$$

In fact, $\log(40) < 0.95 \cdot \log(50)$, so $40^{-d} > (50^{-d})^{0.95}$, so the seed length is at most

$$\begin{aligned} n^{1-\delta} \cdot n^{-0.5\cdot(50^{-d})^{0.95}} \cdot \operatorname{polylog} n &\leq n^{1-\delta} \cdot n^{-\Omega(1/\log^{0.95} n)} \cdot \operatorname{polylog} n \\ &= n^{1-\delta} \cdot 2^{-\Omega(\log^{0.05} n)} \cdot 2^{O(\log\log n)}. \end{aligned}$$

For large enough n, this is indeed at most $n^{1-\delta}$. Explicitness is clear, completing the proof of Theorem 5.3.2.

Chapter 6

Conclusions

The **L** vs. **BPL** problem obviously remains open. It is a challenging problem, but as we have seen, it is possible to make progress on it in various ways. For this reason, derandomizing space-bounded computation continues to be an active area of research, as evidenced by, e.g., Reingold's tutorial on the subject at the DIMACS Day of Complexity Tutorials prior to CCC 2019 [Rei19], or the workshop on the subject at STOC 2020 organized by Meka, Tal, and Zuckerman [MTZ20].

In this dissertation, we have discussed the PRG, HSG, and WPRG approaches to **L** vs. **BPL**. PRGs remain the "gold standard" for derandomization, but the extra flexibility of HSGs and WPRGs means we sometimes have more success constructing them. Of course, some research on **L** vs. **BPL** does not fit neatly into any of these three categories. For example, the recent line of work based on methods from fast Laplacian solvers [MRSV17; MRSV19; AKMPSV20] is probably best thought of as a distinct approach, although it has connections to PRGs and WPRGs (including our PRG for unbounded-width permutation ROBPs and our low-error WPRG for unrestricted ROBPs). Perhaps the next breakthrough will use a completely new derandomization method – but even if so, we can expect work on PRGs, HSGs, and WPRGs to continue to yield valuable insights about ROBPs and **L** vs.

BPL. We conclude with some suggested directions for future research.

6.1 Suggested Open Problems

PRGs for more types of read-once formulas Read-once $\mathbf{AC}^0[\oplus]$ formulas can be simulated by constant-width arbitrary-order ROBPs, and read-once $\mathbf{TC}^0$ formulas can be sandwiched by polynomial-width arbitrary-order ROBPs. Better PRGs for these models would therefore be good progress toward fooling ROBPs. In the read-many setting, $\mathbf{AC}^0[\oplus] \subseteq \mathbf{TC}^0$, but this inclusion does not carry over to the read-once models: read-once $\mathbf{TC}^0$ formulas cannot even compute the parity function on two bits. The problems of fooling read-once $\mathbf{AC}^0[\oplus]$ and fooling read-once $\mathbf{TC}^0$ are incomparable.

Low-error PRGs for permutation ROBPs As we saw in Section 5.1.1, the INW generator [INW94] does a particularly good job of fooling permutation ROBPs. However, even in the constant-width case, the best result is seed length $O(\log n \cdot \log(1/\varepsilon))$ (see Table 5.1). There is much room for improvement when, e.g., $\varepsilon = 1/\operatorname{poly}(n)$. As a starting point, we suggest the problem of designing near-optimal PRGs for *width-3 permutation ROBPs*. This model seems to be a key bottleneck for getting near-optimal PRGs for general width-3 ROBPs [MRT19]. It is a highly restricted model, so perhaps there is a relatively simple solution.

New PRG approaches We currently only know a few general "frameworks" for constructing and analyzing PRGs. For example, there is the Ajtai-Wigderson framework [AW89], which we used to design our PRG for threshold circuits; there is the famous Nisan-Wigderson framework based on hard functions [NW94]; there is the "seed recycling" framework [Nis92; INW94; NZ96], which we used to fool unbounded-width permutation ROBPs; and there is

the recent framework due to Chattopadhyay, Hatami, Hosseini, and Lovett based on polarizing random walks [CHHL19]. It seems likely that we will need a new framework to obtain near-optimal PRGs for ROBPs. It has been suggested that perhaps the bitwise XOR of two small-bias distributions fools ROBPs with near-optimal seed length [LV17], so more methods of analyzing such a construction would be welcome. Another candidate construction is $\mathsf{Res}(X, Y) \circ Z$, where X, Y, and Z are three independent small-bias distributions, i.e., we use Y to pseudorandomly partition $[n]$ into two parts; we use X to assign values to the coordinates in one part and Z to assign values to the coordinates in the other part. Perhaps one of these constructions can at last provide a PRG for read-once CNFs with optimal seed length $O(\log(n/\varepsilon))$. (After considerable research, the best explicit PRG for read-once CNFs currently known has "partially optimal" seed length $O(\log n) + \widetilde{O}(\log(1/\varepsilon))$ [DETT10; GMRTV12; BN17; DHH20].)

BPL vs. NL We suggest trying to prove the implication $\mathbf{L} = \mathbf{NL} \implies \mathbf{L} = \mathbf{BPL}$. That might sound a little silly – what is the point of proving something we expect to be true under an assumption that could go either way? But we think it would be a good first step in a few interesting directions at once. First of all, it would be a step toward proving $\mathbf{BPL} \subseteq \mathbf{NL}$, a problem posed by Saks [Sak96]. Saks further conjectured that the *non-halting* version of **BPL** coincides with **NL** [Sak96], giving extra motivation for proving $\mathbf{BPL} \subseteq \mathbf{NL}$. Second of all, it would be a step toward proving the implication $\mathbf{prL} = \mathbf{prRL} \implies \mathbf{L} = \mathbf{BPL}$, where **prL** and **prRL** are the promise versions of **L** and **RL**. Recall that we showed that optimal HSGs for ROBPs, or equivalently a *black-box* derandomization of **prRL**, would imply $\mathbf{L} = \mathbf{BPL}$ (Section 3.3). It would be great to show that even a *non-black-box* derandomization of **prRL** would imply $\mathbf{L} = \mathbf{BPL}$, matching the known analogous theorem for time-bounded derandomization $\mathbf{prP} = \mathbf{prRP} \implies \mathbf{P} = \mathbf{BPP}$ [BF99]. Finally, this problem might be related to the problem of proving $\mathbf{RL} = \mathbf{coRL}$.

Combining HSGs with the Saks-Zhou framework In Section 3.3, we showed that *optimal* HSGs for ROBPs would imply $\mathbf{L} = \mathbf{BPL}$. What can we achieve if we assume HSGs that are impressive but not optimal? For example, imagine if someone discovered an explicit ε-HSG for width-w length-n ROBPs with seed length $O(\log(wn/\varepsilon) \cdot \sqrt{\log n})$. That would be a huge breakthrough, and yet it is not currently known how to derive any new derandomization of **BPL** or even **RL** from such a construction. A PRG or poly(w)-bounded WPRG with that seed length would imply $\mathbf{BPL} \subseteq \mathbf{DSPACE}(\log^{4/3} N)$ via the Saks-Zhou framework [SZ99]. It would be great to obtain a similar consequence from an HSG. For context, recall that as mentioned in Section 3.3.1, there are many known proofs that optimal HSGs for circuits imply $\mathbf{P} = \mathbf{BPP}$. Those proofs do not all "scale" equally well. That is, some proofs give better results than others under the assumption of explicit HSGs for circuits that are highly nontrivial but far from optimal [GVW11].

More low-error WPRGs and low-threshold HSGs So far, HSGs and WPRGs have been particularly helpful in the small-ε regime. That might continue to be true. As mentioned in Section 4.1, it remains an open problem to construct WPRGs for width-w length-n ROBPs with optimal seed length $O(\log(w/\varepsilon))$ in the regime $n = \operatorname{polylog} w$. Such a WPRG would be similar to the Nisan-Zuckerman PRG [NZ96], but with a better error. An appealing consequence would be that any bounded-error randomized decision algorithm that uses S bits of space and R random bits could be simulated by another bounded-error decision algorithm using $O(S)$ bits of space and R/S^c random bits, where c is an arbitrarily large constant. At the other extreme, we can look at width-3 ROBPs. Near-optimal HSGs for width-3 ROBPs are known [ŠŽ11; GMRTV12], but the best PRG for width-3 ROBPs has seed length $\widetilde{O}(\log n \cdot \log(1/\varepsilon))$ [MRT19]. (For width-4 ROBPs, the best PRG is still Nisan's PRG for polynomial-width ROBPs [Nis92].) Can we design a WPRG for width-3 ROBPs with seed length $\widetilde{O}(\log(n/\varepsilon))$? Finally, it would be great to design HSGs or WPRGs with a

better dependence on ε for *arbitrary-order* ROBPs, compared to the state-of-the-art PRGs by Forbes and Kelley [FK18].

Bibliography

[ACR98] Alexander E. Andreev, Andrea E. F. Clementi, and José D. P. Rolim. "A New General Derandomization Method". In: *J. ACM* 45.1 (1998), pp. 179–213. ISSN: 0004-5411. DOI: 10.1145/273865.273933.

[ACR99] Alexander E. Andreev, Andrea E. F. Clementi, and José D. P. Rolim. "Worst-case hardness suffices for derandomization: a new method for hardness-randomness trade-offs". In: *Theoret. Comput. Sci.* 221.1-2 (1999), pp. 3–18. ISSN: 0304-3975. DOI: 10.1016/S0304-3975(99)00024-9.

[ACRT99] Alexander E. Andreev, Andrea E. F. Clementi, José D. P. Rolim, and Luca Trevisan. "Weak Random Sources, Hitting Sets, and BPP Simulations". In: *SIAM J. Comput.* 28.6 (1999), pp. 2103–2116. ISSN: 0097-5397. DOI: 10.1137/S0097539797325636.

[AGHP92] Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. "Simple Constructions of Almost k-wise Independent Random Variables". In: *Random Structures Algorithms* 3.3 (1992), pp. 289–304. ISSN: 1042-9832. DOI: 10.1002/rsa.3240030308.

[AKLLR79] Romas Aleliunas, Richard M. Karp, Richard J. Lipton, László Lovász, and Charles Rackoff. "Random walks, universal traversal sequences, and the complexity of maze problems". In: *Proceedings of the 20th Symposium on Foundations of Computer Science (FOCS)*. 1979, pp. 218–223. DOI: https://doi.org/10.1109/SFCS.1979.34.

[AKMPSV20] AmirMahdi Ahmadinejad, Jonathan Kelner, Jack Murtagh, John Peebles, Aaron Sidford, and Salil Vadhan. "High-precision Estimation of Random Walks in Small Space". In: *Proceedings of the 61st Symposium on Foundations of Computer Science (FOCS)*. 2020, pp. 1295–1306. DOI: 10.1109/FOCS46700.2020.00123.

[AKS87] Miklós Ajtai, János Komlós, and Endre Szemerédi. "Deterministic Simulation in LOGSPACE". In: *Proceedings of the 19th Symposium on Theory of Computing (STOC)*. 1987, pp. 132–140. ISBN: 0897912217. DOI: 10.1145/28395.28410.

[AM11] Scott Aaronson and Dieter van Melkebeek. "On Circuit Lower Bounds from Derandomization". In: *Theory Comput.* 7 (2011), pp. 177–184. DOI: 10.4086/toc.2011.v007a012.

[And87] A. E. Andreev. "On a method for obtaining more than quadratic effective lower bounds for the complexity of π-schemes". In: *Vestnik Moskov. Univ. Ser. I Mat. Mekh.* 1 (1987), pp. 70–73, 103. ISSN: 0579-9368. URL: http://mi.mathnet.ru/eng/vmumm3034.

[Arm98] Roy Armoni. "On the Derandomization of Space-Bounded Computations". In: *Proceedings of the 2nd International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*. 1998, pp. 47–59. DOI: 10.1007/3-540-49543-6_5.

[ASWZ96] Roy Armoni, Michael Saks, Avi Wigderson, and Shiyu Zhou. "Discrepancy Sets and Pseudorandom Generators for Combinatorial Rectangles". In: *Proceedings of the 37th Annual Symposium on Foundations of Computer Science (FOCS)*. 1996, pp. 412–421. DOI: 10.1109/SFCS.1996.548500.

[AW89] Miklós Ajtai and Avi Wigderson. "Deterministic Simulation of Probabilistic Constant-Depth Circuits". In: *Advances in Computing Research – Randomness and Computation* 5 (1989), pp. 199–23.

[BCG20] Mark Braverman, Gil Cohen, and Sumegha Garg. "Pseudorandom Pseudodistributions with Near-Optimal Error for Read-Once Branching Programs". In: *SIAM Journal on Computing* 49.5 (2020), STOC18–242–STOC18–299. DOI: 10.1137/18M1197734.

[BCP83] A. Borodin, S. Cook, and N. Pippenger. "Parallel Computation for Well-Endowed Rings and Space-Bounded Probabilistic Machines". In: *Inform. and Control* 58.1-3 (1983), pp. 113–136. ISSN: 0019-9958. DOI: 10.1016/S0019-9958(83)80060-6.

[BDS13] Aditya Bhaskara, Devendra Desai, and Srikanth Srinivasan. "Optimal Hitting Sets for Combinatorial Shapes". In: *Theory Comput.* 9 (2013), pp. 441–470. DOI: 10.4086/toc.2013.v009a013.

[BDVY13] Andrej Bogdanov, Zeev Dvir, Elad Verbin, and Amir Yehudayoff. "Pseudorandomness for Width-2 Branching Programs". In: *Theory Comput.* 9 (2013), pp. 283–292. DOI: 10.4086/toc.2013.v009a007.

[BF99] Harry Buhrman and Lance Fortnow. "One-Sided Versus Two-Sided Error in Probabilistic Computation". In: *Proceedings of the 16th Symposium on Theoretical Aspects of Computer Science (STACS)*. 1999, pp. 100–109. DOI: 10.1007/3-540-49116-3_9.

[BGG93] Mihir Bellare, Oded Goldreich, and Shafi Goldwasser. "Randomness in interactive proofs". In: *Comput. Complexity* 3.4 (1993), pp. 319–354. ISSN: 1016-3328. DOI: 10.1007/BF01275487.

[BM84] Manuel Blum and Silvio Micali. "How to Generate Cryptographically Strong Sequences of Pseudorandom Bits". In: *SIAM J. Comput.* 13.4 (1984), pp. 850–864. ISSN: 0097-5397. DOI: 10.1137/0213053.

[BN17] Louay Bazzi and Nagi Nahas. "Small-Bias is Not Enough to Hit Read-Once CNF". In: *Theory of Computing Systems* 60.2 (2017), pp. 324–345. DOI: 10.1007/s00224-016-9680-6.

[BNS92] László Babai, Noam Nisant, and Márió Szegedy. "Multiparty Protocols, Pseudorandom Generators for Logspace, and Time-Space Trade-offs". In: *Journal of Computer and System Sciences* 45.2 (1992), pp. 204–232. ISSN: 0022-0000. DOI: 10.1016/0022-0000(92)90047-M.

[BPW11] Andrej Bogdanov, Periklis A. Papakonstantinou, and Andrew Wan. "Pseudorandomness for Read-Once Formulas". In: *Proceedings of the 52nd Symposium on Foundations of Computer Science (FOCS)*. 2011, pp. 240–246. DOI: 10.1109/FOCS.2011.57.

[BR94] M. Bellare and J. Rompel. "Randomness-Efficient Oblivious Sampling". In: *Proceedings of the 35th Symposium on Foundations of Computer Science (FOCS)*. 1994, pp. 276–287. ISBN: 0818665807. DOI: 10.1109/SFCS.1994.365687.

[Bra10] Mark Braverman. "Polylogarithmic Independence Fools AC^0 Circuits". In: *J. ACM* 57.5 (2010), Art. 28. ISSN: 0004-5411. DOI: 10.1145/1754399.1754401.

[BRRY14] Mark Braverman, Anup Rao, Ran Raz, and Amir Yehudayoff. "Pseudorandom Generators for Regular Branching Programs". In: *SIAM J. Comput.* 43.3 (2014), pp. 973–986. ISSN: 0097-5397. DOI: 10.1137/120875673.

[BV10] Joshua Brody and Elad Verbin. "The Coin Problem and Pseudorandomness for Branching Programs". In: *Proceedings of the 51st Symposium on Foundations of Computer Science (FOCS)*. 2010, pp. 30–39. DOI: 10.1109/FOCS.2010.10. URL: https://www.cs.swarthmore.edu/~brody/papers/CoinProblemFullVersion.pdf.

[CDRSTS21] Gil Cohen, Dean Doron, Oren Renard, Ori Sberlo, and Amnon Ta-Shma. *Error Reduction For Weighted PRGs Against Read Once Branching Programs*. 2021. ECCC: TR21-020.

[CDS19] Eshan Chattopadhyay, Anindya De, and Rocco A. Servedio. "Simple and Efficient Pseudorandom Generators from Gaussian Processes". In: *Proceedings of the 34th Computational Complexity Conference (CCC)*. 2019, Art. 4. DOI: 10.4230/LIPIcs.CCC.2019.4.

[CH20] Kuan Cheng and William M. Hoza. "Hitting Sets Give Two-Sided Derandomization of Small Space". In: *Proceedings of the 35th Computational Complexity Conference (CCC)*. 2020, 10:1–10:25. ISBN: 978-3-95977-156-6. DOI: 10.4230/LIPIcs.CCC.2020.10.

[CHHL19] Eshan Chattopadhyay, Pooya Hatami, Kaave Hosseini, and Shachar Lovett. "Pseudorandom Generators from Polarizing Random Walks". In: *Theory Comput.* 15 (2019), Paper No. 10. DOI: 10.4086/toc.2019.v015a010.

[CHMY21] Mahdi Cheraghchi, Shuichi Hirahara, Dimitrios Myrisiotis, and Yuichi Yoshida. "One-Tape Turing Machine and Branching Program Lower Bounds for MCSP". In: *Proceedings of the 38th International Symposium on Theoretical Aspects of Computer Science (STACS)*. 2021, 23:1–23:19. ISBN: 978-3-95977-180-1. DOI: 10.4230/LIPIcs.STACS.2021.23.

[CHRT18] Eshan Chattopadhyay, Pooya Hatami, Omer Reingold, and Avishay Tal. "Improved Pseudorandomness for Unordered Branching Programs through Local Monotonicity". In: *Proceedings of the 50th Symposium on Theory of Computing (STOC)*. 2018, pp. 363–375. DOI: 10.1145/3188745.3188800.

[CIKK15] Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. "Tighter Connections between Derandomization and Circuit Lower Bounds". In: *Proceedings of the 19th International Workshop on Randomization and Computation (RANDOM)*. 2015, pp. 645–658. DOI: 10.4230/LIPIcs.APPROX-RANDOM.2015.645.

[CKPPRSV17] Michael B. Cohen, Jonathan Kelner, John Peebles, Richard Peng, Anup B. Rao, Aaron Sidford, and Adrian Vladu. "Almost-linear-time algorithms for Markov chains and new spectral primitives for directed graphs". In: *Proceedings of the 49th Symposium on Theory of Computing (STOC)*. 2017, pp. 410–419. DOI: 10.1145/3055399.3055463.

[CL20] Eshan Chattopadhyay and Jyun-Jie Liao. "Optimal Error Pseudodistributions for Read-Once Branching Programs". In: *Proceedings of the 35th Computational Complexity Conference (CCC)*. 2020, 25:1–25:27. ISBN: 978-3-95977-156-6. DOI: 10.4230/LIPIcs.CCC.2020.25.

[CL21] Lijie Chen and Xin Lyu. *Inverse-Exponential Correlation Bounds and Extremely Rigid Matrices from a New Derandomized XOR Lemma*. 2021. ECCC: TR21-003.

[CRS00] Suresh Chari, Pankaj Rohatgi, and Aravind Srinivasan. "Improved Algorithms via Approximations of Probability Distributions". In: *J. Comput. System Sci.* 61.1 (2000), pp. 81–107. ISSN: 0022-0000. DOI: 10.1006/jcss.1999.1695.

[CSS18] Ruiwen Chen, Rahul Santhanam, and Srikanth Srinivasan. "Average-Case Lower Bounds and Satisfiability Algorithms for Small Threshold Circuits". In: *Theory Comput.* 14 (2018), Paper No. 9, 55. DOI: 10.4086/toc.2018.v014a009.

[CSV15] Sitan Chen, Thomas Steinke, and Salil Vadhan. *Pseudorandomness for Read-Once, Constant-Depth Circuits*. 2015. arXiv: 1504.04675.

[CT20] Lijie Chen and Roei Tell. *Simple and fast derandomization from very hard functions: Eliminating randomness at almost no cost*. 2020. ECCC: TR20-148.

[De11] Anindya De. "Pseudorandomness for Permutation and Regular Branching Programs". In: *Proceedings of the 26th Conference on Computational Complexity (CCC)*. 2011, pp. 221–231. DOI: 10.1109/CCC.2011.23.

[De15] Anindya De. "Beyond the Central Limit theorem: Asymptotic Expansions and Pseudorandomness for Combinatorial Sums". In: *56th Symposium on Foundations of Computer Science (FOCS)*. 2015, pp. 883–902. DOI: 10.1109/FOCS.2015.59.

[DETT10] Anindya De, Omid Etesami, Luca Trevisan, and Madhur Tulsiani. "Improved Pseudorandom Generators for Depth 2 Circuits". In: *Proceedings of the 14th International Workshop on Randomization and Computation (RANDOM)*. 2010, pp. 504–517. DOI: 10.1007/978-3-642-15369-3_38.

[DGJSV10] Ilias Diakonikolas, Parikshit Gopalan, Ragesh Jaiswal, Rocco A. Servedio, and Emanuele Viola. "Bounded Independence Fools Halfspaces". In: *SIAM J. Comput.* 39.8 (2010), pp. 3441–3462. ISSN: 0097-5397. DOI: 10.1137/100783030.

[DHH19] Dean Doron, Pooya Hatami, and William M. Hoza. "Near-Optimal Pseudorandom Generators for Constant-Depth Read-Once Formulas". In: *Proceedings of the 34th Computational Complexity Conference (CCC)*. 2019, 16:1–16:34. DOI: 10.4230/LIPIcs.CCC.2019.16.

[DHH20] Dean Doron, Pooya Hatami, and William M. Hoza. "Log-Seed Pseudorandom Generators via Iterated Restrictions". In: *Proceedings of the 35th Computational Complexity Conference (CCC)*. 2020, 6:1–6:36. ISBN: 978-3-95977-156-6. DOI: 10.4230/LIPIcs.CCC.2020.6.

[DMOZ20] Dean Doron, Dana Moshkovitz, Justin Oh, and David Zuckerman. "Nearly Optimal Pseudorandomness From Hardness". In: *61st Symposium on Foundations of Computer Science (FOCS)*. 2020, pp. 1057–1068. DOI: 10.1109/FOCS46700.2020.00102.

[DMRTV21] Dean Doron, Raghu Meka, Omer Reingold, Avishay Tal, and Salil Vadhan. *Pseudorandom Generators for Read-Once Monotone Branching Programs*. 2021. ECCC: TR21-018.

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. "Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data". In: *SIAM Journal on Computing* 38.1 (2008), pp. 97–139. ISSN: 0097-5397. DOI: 10.1137/060651380.

[DPS11] Matei David, Periklis A. Papakonstantinou, and Anastasios Sidiropoulos. "How strong is Nisan's pseudo-random generator?" In: *Inform. Process. Lett.* 111.16 (2011), pp. 804–808. ISSN: 0020-0190. DOI: 10.1016/j.ipl.2011.04.013.

[DTST20] Dean Doron, Amnon Ta-Shma, and Roei Tell. "On Hitting-Set Generators for Polynomials That Vanish Rarely". In: *Proceedings of the 24th International Conference on Randomization and Computation (RANDOM)*. 2020, 7:1–7:23. ISBN: 978-3-95977-164-1. DOI: 10.4230/LIPIcs.APPROX/RANDOM.2020.7.

[DZ94] Moshe Dubiner and Uri Zwick. "How Do Read-Once Formulae Shrink?" In: *Combin. Probab. Comput.* 3.4 (1994), pp. 455–469. ISSN: 0963-5483. DOI: 10.1017/S0963548300001358.

[EGLNV98] Guy Even, Oded Goldreich, Michael Luby, Noam Nisan, and Boban Veličković. "Efficient Approximation of Product Distributions". In: *Random Structures Algorithms* 13.1 (1998), pp. 1–16. ISSN: 1042-9832. DOI: 10.1002/(SICI)1098-2418(199808)13:1<1::AID-RSA1>3.0.CO;2-W.

[Erd45] P. Erdős. “On a lemma of Littlewood and Offord”. In: *Bull. Amer. Math. Soc.* 51 (1945), pp. 898–902. ISSN: 0002-9904. DOI: 10.1090/S0002-9904-1945-08454-7.

[FK18] Michael A. Forbes and Zander Kelley. “Pseudorandom Generators for Read-Once Branching Programs, in Any Order”. In: *Proceedings of the 59th Symposium on Foundations of Computer Science (FOCS)*. 2018, pp. 946–955. DOI: 10.1109/FOCS.2018.00093.

[Fre81] Rūsiņš Freivalds. “Probabilistic two-way machines”. In: *Proceedings of the 10th International Symposium on Mathematical Foundations of Computer Science (MFCS)*. 1981, pp. 33–45. DOI: 10.1007/3-540-10856-4_72.

[Gil77] John Gill. “Computational Complexity of Probabilistic Turing Machines”. In: *SIAM J. Comput.* 6.4 (1977), pp. 675–695. ISSN: 0097-5397. DOI: 10.1137/0206049.

[GKM18] Parikshit Gopalan, Daniel M. Kane, and Raghu Meka. “Pseudorandomness via the Discrete Fourier Transform”. In: *SIAM J. Comput.* 47.6 (2018), pp. 2451–2487. ISSN: 0097-5397. DOI: 10.1137/16M1062132.

[GLS12] Dmitry Gavinsky, Shachar Lovett, and Srikanth Srinivasan. “Pseudorandom Generators for Read-Once ACC^0”. In: *Proceedings of the 27th Conference on Computational Complexity (CCC)*. 2012, pp. 287–297. DOI: 10.1109/CCC.2012.37.

[GLSS15] Dmitry Gavinsky, Shachar Lovett, Michael Saks, and Srikanth Srinivasan. “A Tail Bound for Read-k Families of Functions”. In: *Random Structures Algorithms* 47.1 (2015), pp. 99–108. ISSN: 1042-9832. DOI: 10.1002/rsa.20532.

[GMRTV12] Parikshit Gopalan, Raghu Meka, Omer Reingold, Luca Trevisan, and Salil Vadhan. “Better Pseudorandom Generators from Milder Pseudorandom Restrictions”. In: *Proceedings of the 53rd Symposium on Foundations of Computer Science (FOCS)*. 2012, pp. 120–129. DOI: 10.1109/FOCS.2012.77.

[GMRZ13] Parikshit Gopalan, Raghu Meka, Omer Reingold, and David Zuckerman. “Pseudorandom Generators for Combinatorial Shapes”. In: *SIAM J. Comput.* 42.3 (2013), pp. 1051–1076. ISSN: 0097-5397. DOI: 10.1137/110854990.

[Gol11] Oded Goldreich. "A Sample of Samplers: A Computational Perspective on Sampling". In: *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*. Vol. 6650. Lecture Notes in Comput. Sci. Springer, Heidelberg, 2011, pp. 302–332. DOI: 10.1007/978-3-642-22670-0_24.

[GOWZ10] Parikshit Gopalan, Ryan O'Donnell, Yi Wu, and David Zuckerman. "Fooling Functions of Halfspaces under Product Distributions". In: *Proceedings of the 25th IEEE Conference on Computational Complexity (CCC)*. 2010, pp. 223–234. DOI: 10.1109/CCC.2010.29.

[GV20] Rohit Gurjar and Ben Lee Volk. "Pseudorandom Bits for Oblivious Branching Programs". In: *ACM Trans. Comput. Theory* 12.2 (2020), Art. 8. ISSN: 1942-3454. DOI: 10.1145/3378663.

[GVW11] Oded Goldreich, Salil Vadhan, and Avi Wigderson. "Simplified Derandomization of BPP Using a Hitting Set Generator". In: *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*. Vol. 6650. Lecture Notes in Comput. Sci. Springer, Heidelberg, 2011, pp. 59–67. DOI: 10.1007/978-3-642-22670-0_8.

[GY20] Parikshit Gopalan and Amir Yehudayoff. "Concentration for Limited Independence via Inequalities for the Elementary Symmetric Polynomials". In: *Theory of Computing* 16.17 (2020), pp. 1–29. DOI: 10.4086/toc.2020.v016a017.

[Hås14] Johan Håstad. "On the Correlation of Parity and Small-Depth Circuits". In: *SIAM J. Comput.* 43.5 (2014), pp. 1699–1708. ISSN: 0097-5397. DOI: 10.1137/120897432.

[Hås98] Johan Håstad. "The Shrinkage Exponent of de Morgan Formulas is 2". In: *SIAM J. Comput.* 27.1 (1998), pp. 48–64. ISSN: 0097-5397. DOI: 10.1137/S0097539794261556.

[HHR11] Iftach Haitner, Danny Harnik, and Omer Reingold. "On the Power of the Randomized Iterate". In: *SIAM J. Comput.* 40.6 (2011), pp. 1486–1528. ISSN: 0097-5397. DOI: 10.1137/080721820.

[HHTT21] Pooya Hatami, William M. Hoza, Avishay Tal, and Roei Tell. *Fooling Constant-Depth Threshold Circuits*. 2021. ECCC: TR21-002.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. "A pseudorandom generator from any one-way function". In: *SIAM J. Comput.* 28.4 (1999), pp. 1364–1396. ISSN: 0097-5397. DOI: 10.1137/S0097539793244708.

[HKM12] Prahladh Harsha, Adam Klivans, and Raghu Meka. "An Invariance Principle for Polytopes". In: *J. ACM* 59.6 (2012), Art. 29. ISSN: 0004-5411. DOI: 10.1145/2395116.2395118.

[HLV18] Elad Haramaty, Chin Ho Lee, and Emanuele Viola. "Bounded Independence Plus Noise Fools Products". In: *SIAM J. Comput.* 47.2 (2018), pp. 493–523. ISSN: 0097-5397. DOI: 10.1137/17M1129088.

[Hoz19a] William Hoza. *Near-Optimal Pseudorandom Generators for Constant-Depth Read-Once Formulas (oral presentation)*. Presented at Banff International Research Station workshop 19w5088. 2019. URL: http://www.birs.ca/events/2019/5-day-workshops/19w5088/videos/watch/201907081331-Hoza.html.

[Hoz19b] William M. Hoza. "Typically-Correct Derandomization for Small Time and Space". In: *Proceedings of the 34th Computational Complexity Conference (CCC)*. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2019, 9:1–9:39. DOI: 10.4230/LIPIcs.CCC.2019.9.

[Hoz20] William Hoza. *Hitting Sets Give Two-Sided Derandomization of Small Space (oral presentation)*. Presented at Computational Complexity Conference. 2020. URL: https://www.youtube.com/watch?v=5CX1qaF5q4A.

[Hoz21a] William Hoza. *Fooling Constant-Depth Threshold Circuits (oral presentation)*. Presented at the TCS+ Seminar. 2021. URL: https://www.youtube.com/watch?v=xZi1OIu3riQ.

[Hoz21b] William M. Hoza. *Better Pseudodistributions and Derandomization for Space-Bounded Computation*. 2021. ECCC: TR21-048.

[HPV21] William M. Hoza, Edward Pyne, and Salil Vadhan. "Pseudorandom Generators for Unbounded-Width Permutation Branching Programs". In: *Proceedings of the 12th Innovations in Theoretical Computer Science Conference (ITCS)*. 2021, 7:1–7:20. ISBN: 978-3-95977-177-1. DOI: 10.4230/LIPIcs.ITCS.2021.7.

[HRY95] Johan Håstad, Alexander Razborov, and Andrew Yao. "On the shrinkage exponent for read-once formulae". In: *Theoret. Comput. Sci.* 141.1-2 (1995), pp. 269–282. ISSN: 0304-3975. DOI: 10.1016/0304-3975(94)00081-S.

[HS19] Prahladh Harsha and Srikanth Srinivasan. "On Polynomial Approximations to AC^0". In: *Random Structures Algorithms* 54.2 (2019), pp. 289–303. ISSN: 1042-9832. DOI: 10.1002/rsa.20786.

[HU21] William M. Hoza and Chris Umans. “Targeted Pseudorandom Generators, Simulation Advice Generators, and Derandomizing Logspace”. In: *SIAM Journal on Computing* (2021), STOC17–281–STOC17–304. DOI: 10.1137/17M1145707.

[HVV06] Alexander Healy, Salil Vadhan, and Emanuele Viola. “Using Nondeterminism to Amplify Hardness”. In: *SIAM J. Comput.* 35.4 (2006), pp. 903–931. ISSN: 0097-5397. DOI: 10.1137/S0097539705447281.

[HZ20] William M. Hoza and David Zuckerman. “Simple Optimal Hitting Sets for Small-Success RL”. In: *SIAM Journal on Computing* 49.4 (2020), pp. 811–820. DOI: 10.1137/19M1268707.

[IMZ19] Russell Impagliazzo, Raghu Meka, and David Zuckerman. “Pseudorandomness from Shrinkage”. In: *J. ACM* 66.2 (2019), Art. 11. ISSN: 0004-5411. DOI: 10.1145/3230630.

[IN93] Russell Impagliazzo and Noam Nisan. “The Effect of Random Restrictions on Formula Size”. In: *Random Structures Algorithms* 4.2 (1993), pp. 121–133. ISSN: 1042-9832. DOI: 10.1002/rsa.3240040202.

[Ind06] Piotr Indyk. “Stable Distributions, Pseudorandom Generators, Embeddings, and Data Stream Computation”. In: *J. ACM* 53.3 (2006), pp. 307–323. ISSN: 0004-5411. DOI: 10.1145/1147954.1147955.

[INW94] Russell Impagliazzo, Noam Nisan, and Avi Wigderson. “Pseudorandomness for Network Algorithms”. In: *Proceedings of the 26th Symposium on Theory of Computing (STOC)*. 1994, pp. 356–364. ISBN: 0897916638. DOI: 10.1145/195058.195190.

[IPS97] Russell Impagliazzo, Ramamohan Paturi, and Michael E. Saks. “Size-Depth Tradeoffs for Threshold Circuits”. In: *SIAM J. Comput.* 26.3 (1997), pp. 693–707. ISSN: 0097-5397. DOI: 10.1137/S0097539792282965.

[ISW99] Russell Impagliazzo, Ronen Shaltiel, and Avi Wigderson. “Near-Optimal conversion of Hardness into Pseudo-Randomness”. In: *Proceedings of the 40th Symposium on Foundations of Computer Science (FOCS)*. 1999, pp. 181–190. DOI: 10.1109/SFFCS.1999.814590.

[IW97] Russell Impagliazzo and Avi Wigderson. “P = BPP If E Requires Exponential Circuits: Derandomizing the XOR Lemma”. In: *Proceedings of the 29th Symposium on Theory of Computing (STOC)*. 1997, pp. 220–229. ISBN: 0897918886. DOI: 10.1145/258533.258590.

[JS12] Maurice Jansen and Rahul Santhanam. "Stronger Lower Bounds and Randomness-Hardness Trade-Offs Using Associated Algebraic Complexity Classes". In: *Proceedings of the 29th International Symposium on Theoretical Aspects of Computer Science (STACS)*. 2012, pp. 519–530. DOI: 10.4230/LIPIcs.STACS.2012.519.

[Jun81] H. Jung. "Relationships between probabilistic and deterministic tape complexity". In: *Proceedings of the 10th Symposium on Mathematical Foundations of Computer Science (MFCS)*. 1981, pp. 339–346. DOI: 10.1007/3-540-10856-4_101.

[Jun84] Hermann Jung. "On probabilistic tape complexity and fast circuits for matrix inversion problems". In: *Proceedings of the 11th International Colloquium on Automata, Languages and Programming (ICALP)*. 1984, pp. 281–291. ISBN: 978-3-540-38886-9. DOI: 10.1007/3-540-13345-3_25.

[Kel20] Zander Kelley. *An Improved Derandomization of the Switching Lemma*. 2020. ECCC: TR20-182.

[KI04] Valentine Kabanets and Russell Impagliazzo. "Derandomizing Polynomial Identity Tests Means Proving Circuit Lower Bounds". In: *Comput. Complexity* 13.1-2 (2004), pp. 1–46. ISSN: 1016-3328. DOI: 10.1007/s00037-004-0182-6.

[KKLMO20] Valentine Kabanets, Sajin Koroth, Zhenjian Lu, Dimitrios Myrisiotis, and Igor C. Oliveira. "Algorithms and Lower Bounds for De Morgan Formulas of Low-Communication Leaf Gates". In: *Proceedings of the 35th Computational Complexity Conference (CCC)*. 2020, 15:1–15:41. ISBN: 978-3-95977-156-6. DOI: 10.4230/LIPIcs.CCC.2020.15.

[KLW10] Adam R. Klivans, Homin K. Lee, and Andrew Wan. "Mansour's Conjecture is True for Random DNF Formulas". In: *Proceedings of the 23rd Conference on Learning Theory (COLT)*. Ed. by Adam Tauman Kalai and Mehryar Mohri. 2010, pp. 368–380. URL: http://www.learningtheory.org/colt2010/papers/085Lee.pdf.

[KM02] Adam R. Klivans and Dieter van Melkebeek. "Graph Nonisomorphism Has Subexponential Size Proofs Unless the Polynomial-Time Hierarchy Collapses". In: *SIAM J. Comput.* 31.5 (2002), pp. 1501–1526. ISSN: 0097-5397. DOI: 10.1137/S0097539700389652.

[KNP11] Michal Koucký, Prajakta Nimbhorkar, and Pavel Pudlák. "Pseudorandom Generators for Group Products". In: *Proceedings of the 43rd Symposium on Theory of Computing (STOC)*. 2011, pp. 263–272. DOI: 10.1145/1993636.1993672.

[KNW08] Daniel M. Kane, Jelani Nelson, and David P. Woodruff. *Revisiting Norm Estimation in Data Streams*. 2008. arXiv: 0811.3648.

[KV85] Marek Karpinski and Rutger Verbeek. "There Is No Polynomial Deterministic Space Simulation of Probabilistic Space with a Two-Way Random-Tape Generator". In: *Inform. and Control* 67.1-3 (1985), pp. 158–162. ISSN: 0019-9958. DOI: 10.1016/S0019-9958(85)80032-2.

[KV87] Marek Karpinski and Rutger Verbeek. "On the Monte Carlo Space Constructible Functions and Separation Results for Probabilistic Complexity Classes". In: *Inform. and Comput.* 75.2 (1987), pp. 178–189. ISSN: 0890-5401. DOI: 10.1016/0890-5401(87)90057-5.

[Lee19] Chin Ho Lee. "Fourier Bounds and Pseudorandom Generators for Product Tests". In: *Proceedings of the 34th Computational Complexity Conference (CCC)*. 2019, 7:1–7:25. DOI: 10.4230/LIPIcs.CCC.2019.7.

[LLSZ97] Nathan Linial, Michael Luby, Michael Saks, and David Zuckerman. "Efficient construction of a small hitting set for combinatorial rectangles in high dimension". In: *Combinatorica* 17.2 (1997), pp. 215–234. ISSN: 0209-9683. DOI: 10.1007/BF01200907.

[LO43] J. E. Littlewood and A. C. Offord. "On the number of real roots of a random algebraic equation. III". In: *Rec. Math. [Mat. Sbornik] N.S.* 12(54) (1943), pp. 277–286. URL: http://mi.mathnet.ru/eng/msb6161.

[LS11] Shachar Lovett and Srikanth Srinivasan. "Correlation Bounds for Poly-size AC^0 Circuits with $n^{1-o(1)}$ Symmetric Gates". In: *Proceedings of the 15th International Workshop on Randomization and Computation (RANDOM)*. 2011, pp. 640–651. DOI: 10.1007/978-3-642-22935-0_54.

[Lu02] Chi-Jen Lu. "Improved Pseudorandom Generators for Combinatorial Rectangles". In: *Combinatorica* 22.3 (2002), pp. 417–433. ISSN: 0209-9683. DOI: 10.1007/s004930200021.

[Lu12] Chi-Jen Lu. "Hitting Set Generators for Sparse Polynomials over Any Finite Fields". In: *Proceedings of the 27th Conference on Computational Complexity (CCC)*. 2012, pp. 280–286. DOI: 10.1109/CCC.2012.20.

[LV17] Chin Ho Lee and Emanuele Viola. “Some Limitations of the Sum of Small-Bias Distributions”. In: *Theory Comput.* 13 (2017), Paper No. 16. DOI: 10.4086/toc.2017.v013a016.

[LV96] M. Luby and B. Veličković. “On Deterministic Approximation of DNF”. In: *Algorithmica* 16.4 (1996), pp. 415–433. DOI: 10.1007/BF01940873.

[Mac98] Ioan I. Macarie. “Space-Efficient Deterministic Simulation of Probabilistic Automata”. In: *SIAM J. Comput.* 27.2 (1998), pp. 448–465. ISSN: 0097-5397. DOI: 10.1137/S0097539793253851.

[Mic92] Pascal Michel. “A survey of space complexity”. In: *Theoret. Comput. Sci.* 101.1 (1992), pp. 99–132. ISSN: 0304-3975. DOI: 10.1016/0304-3975(92)90151-5.

[MPV15] Debasis Mandal, A. Pavan, and N. V. Vinodchandran. “On Probabilistic Space-Bounded Machines with Multiple Access to Random Tape”. In: *Proceedings of the 40th International Symposium on Mathematical Foundations of Computer Science (MFCS)*. Berlin, Heidelberg, 2015, pp. 459–471. ISBN: 978-3-662-48054-0. DOI: 10.1007/978-3-662-48054-0_38.

[MRSV17] Jack Murtagh, Omer Reingold, Aaron Sidford, and Salil Vadhan. “Derandomization Beyond Connectivity: Undirected Laplacian Systems in Nearly Logarithmic Space”. In: *Proceedings of the 58th Symposium on Foundations of Computer Science (FOCS)*. 2017, pp. 801–812. DOI: 10.1109/FOCS.2017.79.

[MRSV19] Jack Murtagh, Omer Reingold, Aaron Sidford, and Salil Vadhan. “Deterministic Approximation of Random Walks in Small Space”. In: *Proceedings of the 23rd International Conference on Randomization and Computation (RANDOM)*. 2019, 42:1–42:22. DOI: 10.4230/LIPIcs.APPROX-RANDOM.2019.42.

[MRT19] Raghu Meka, Omer Reingold, and Avishay Tal. “Pseudorandom Generators for Width-3 Branching Programs”. In: *Proceedings of the 51st Symposium on Theory of Computing (STOC)*. 2019, pp. 626–637. DOI: 10.1145/3313276.3316319.

[MS82] Burkhard Monien and Ivan Hal Sudborough. “On eliminating nondeterminism from turing machines which use less than logarithm worktape space”. In: *Theoret. Comput. Sci.* 21.3 (1982), pp. 237–253. ISSN: 0304-3975. DOI: 10.1016/0304-3975(82)90075-5.

[MTZ20] Raghu Meka, Avishay Tal, and David Zuckerman. *Derandomizing Space-Bounded Computation*. Workshop at STOC 2020. 2020. URL: https://sites.google.com/site/avishaytal/derandomizing-space-bounded-computation.

[MV05] Peter Bro Miltersen and N. V. Vinodchandran. "Derandomizing Arthur-Merlin games using Hitting Sets". In: *Comput. Complexity* 14.3 (2005), pp. 256–279. ISSN: 1016-3328. DOI: 10.1007/s00037-005-0197-7.

[MZ13] Raghu Meka and David Zuckerman. "Pseudorandom Generators for Polynomial Threshold Functions". In: *SIAM J. Comput.* 42.3 (2013), pp. 1275–1301. ISSN: 0097-5397. DOI: 10.1137/100811623.

[Nis91] Noam Nisan. "Pseudorandom bits for constant depth circuits". In: *Combinatorica* 11.1 (1991), pp. 63–70. ISSN: 0209-9683. DOI: 10.1007/BF01375474.

[Nis92] Noam Nisan. "Pseudorandom generators for space-bounded computation". In: *Combinatorica* 12.4 (1992), pp. 449–461. ISSN: 0209-9683. DOI: 10.1007/BF01305237.

[Nis93a] Noam Nisan. "On read-once vs. multiple access to randomness in logspace". In: *Theoret. Comput. Sci.* 107.1 (1993), pp. 135–144. ISSN: 0304-3975. DOI: 10.1016/0304-3975(93)90258-U.

[Nis93b] Noam Nisan. "The communication complexity of threshold gates". In: *Combinatorics, Paul Erdős is eighty, Vol. 1*. Bolyai Soc. Math. Stud. János Bolyai Math. Soc., Budapest, 1993, pp. 301–315. URL: https://www.cs.huji.ac.il/~noam/thresh.ps.

[Nis94] Noam Nisan. "RL $\subseteq$ SC". In: *Comput. Complexity* 4.1 (1994), pp. 1–11. ISSN: 1016-3328. DOI: 10.1007/BF01205052.

[NN93] Joseph Naor and Moni Naor. "Small-Bias Probability Spaces: Efficient Constructions and Applications". In: *SIAM J. Comput.* 22.4 (1993), pp. 838–856. ISSN: 0097-5397. DOI: 10.1137/0222053.

[NW94] Noam Nisan and Avi Wigderson. "Hardness vs Randomness". In: *J. Comput. System Sci.* 49.2 (1994), pp. 149–167. ISSN: 0022-0000. DOI: 10.1016/S0022-0000(05)80043-1.

[NZ96] Noam Nisan and David Zuckerman. "Randomness is Linear in Space". In: *J. Comput. System Sci.* 52.1 (1996), pp. 43–52. ISSN: 0022-0000. DOI: 10.1006/jcss.1996.0004.

[O'D14] Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, New York, 2014. ISBN: 978-1-107-03832-5. DOI: 10.1017/CBO9781139814782.

[OST19] Ryan O'Donnell, Rocco A. Servedio, and Li-Yang Tan. "Fooling Polytopes". In: *Proceedings of the 51st Symposium on Theory of Computing (STOC)*. 2019, pp. 614–625. DOI: 10.1145/3313276.3316321.

[Per90] Rene Peralta. *On the randomness complexity of algorithms*. CS Research Report TR90-1. University of Wisconsin, Milwaukee, 1990.

[PV21] Edward Pyne and Salil Vadhan. *Pseudodistributions That Beat All Pseudorandom Generators*. 2021. ECCC: TR21-019.

[Pyn21] Edward Pyne. *Pseudorandom Generators for Unbounded-Width Permutation Branching Programs (oral presentation)*. Presented at Innovations in Theoretical Computer Science Conference. 2021. URL: https://www.youtube.com/watch?v=CqigKXo7qYo.

[PZ93] Michael S. Paterson and Uri Zwick. "Shrinkage of De Morgan Formulae under Restriction". In: *Random Structures Algorithms* 4.2 (1993), pp. 135–150. ISSN: 1042-9832. DOI: 10.1002/rsa.3240040203.

[Rei08] Omer Reingold. "Undirected Connectivity in Log-Space". In: *J. ACM* 55.4 (2008), Art. 17, 24. ISSN: 0004-5411. DOI: 10.1145/1391289.1391291.

[Rei10] Omer Reingold. *Randomness vs. Memory: Prospects and Barriers (oral presentation)*. Presented at "Barriers in Computational Complexity" workshop. 2010. URL: https://omereingold.files.wordpress.com/2014/10/rlbarriers.pptx.

[Rei19] Omer Reingold. *Recent Developments Related to RL vs. L*. Tutorial at the DIMACS Day of Complexity Tutorials. 2019. URL: http://dimacs.rutgers.edu/events/details?eID=1192.

[RS10] Yuval Rabani and Amir Shpilka. "Explicit Construction of a Small ε-Net for Linear Threshold Functions". In: *SIAM J. Comput.* 39.8 (2010), pp. 3501–3520. ISSN: 0097-5397. DOI: 10.1137/090764190.

[RSV13] Omer Reingold, Thomas Steinke, and Salil Vadhan. "Pseudorandomness for Regular Branching Programs via Fourier Analysis". In: *Proceedings of the 17th International Workshop on Randomization and Computation (RANDOM)*. 2013, pp. 655–670. DOI: 10.1007/978-3-642-40328-6_45.

[RTV06] Omer Reingold, Luca Trevisan, and Salil Vadhan. "Pseudorandom Walks on Regular Digraphs and the **RL** vs. **L** Problem". In: *Proceedings of the 38th Symposium on Theory of Computing (STOC)*. 2006, pp. 457–466. DOI: 10.1145/1132516.1132583.

[RV05] Eyal Rozenman and Salil Vadhan. "Derandomized Squaring of Graphs". In: *Proceedings of the 9th International Workshop on Randomization and Computation (RANDOM)*. 2005, pp. 436–447. DOI: 10.1007/11538462_37.

[Sak96] M. Saks. "Randomization and Derandomization in Space-Bounded Computation". In: *Proceedings of the 11th Conference on Computational Complexity (CCC)*. 1996, pp. 128–149. DOI: 10.1109/CCC.1996.507676.

[Sav70] Walter J. Savitch. "Relationships Between Nondeterministic and Deterministic Tape Complexities". In: *J. Comput. System Sci.* 4 (1970), pp. 177–192. ISSN: 0022-0000. DOI: 10.1016/S0022-0000(70)80006-X.

[Sch03] Alexander Schrijver. *Combinatorial Optimization. Polyhedra and Efficiency. Vol. A*. Vol. 24. Algorithms and Combinatorics. Paths, flows, matchings, Chapters 1–38. Springer-Verlag, Berlin, 2003. ISBN: 3-540-44389-4.

[She13] Alexander A. Sherstov. "Making Polynomials Robust to Noise". In: *Theory Comput.* 9 (2013), pp. 593–615. DOI: 10.4086/toc.2013.v009a018.

[SHL65] R. E. Stearns, J. Hartmanis, and P. M. Lewis. "Hierarchies of memory limited computations". In: *Proceedings of the 6th Symposium on Switching Circuit Theory and Logical Design (SWCT)*. 1965, pp. 179–190. DOI: 10.1109/FOCS.1965.11.

[Siv02] D. Sivakumar. "Algorithmic derandomization via complexity theory". In: *Proceedings of the 34th Symposium on Theory of Computing (STOC)*. 2002, pp. 619–626. DOI: 10.1145/509907.509996.

[ST11] Daniel A. Spielman and Shang-Hua Teng. "Spectral sparsification of graphs". In: *SIAM J. Comput.* 40.4 (2011), pp. 981–1025. ISSN: 0097-5397. DOI: 10.1137/08074489X.

[ST17a] Rocco Servedio and Li-Yang Tan. "Learning and fooling depth-two threshold circuits". Unpublished. 2017.

[ST17b] Rocco A. Servedio and Li-Yang Tan. "Fooling Intersections of Low-Weight Halfspaces". In: *Proceedings of the 58th Symposium on Foundations of Computer Science (FOCS)*. 2017, pp. 824–835. DOI: 10.1109/FOCS.2017.81.

[ST18] Rocco A. Servedio and Li-Yang Tan. "Luby-Veličković-Wigderson Revisited: Improved Correlation Bounds and Pseudorandom Generators for Depth-Two Circuits". In: *Proceedings of the 22nd International Conference on Randomization and Computation (RANDOM)*. 2018, Art. No. 56. DOI: 10.4230/LIPIcs.APPROX-RANDOM.2018.56.

[ST19a] Rocco A. Servedio and Li-Yang Tan. "Improved Pseudorandom Generators from Pseudorandom Multi-Switching Lemmas". In: *Proceedings of the 23rd International Conference on Randomization and Computation (RANDOM)*. 2019, Art. No. 45. DOI: 10.4230/LIPIcs.APPROX-RANDOM.2019.45.

[ST19b] Rocco A. Servedio and Li-Yang Tan. "Pseudorandomness for read-k DNF formulas". In: *Proceedings of the 30th Symposium on Discrete Algorithms (SODA)*. 2019, pp. 621–638. DOI: 10.1137/1.9781611975482.39.

[Ste12] Thomas Steinke. *Pseudorandomness for Permutation Branching Programs Without the Group Theory*. 2012. ECCC: TR12-083.

[SU05] Ronen Shaltiel and Christopher Umans. "Simple Extractors for All Min-Entropies and a New Pseudorandom Generator". In: *J. ACM* 52.2 (2005), pp. 172–216. ISSN: 0004-5411. DOI: 10.1145/1059513.1059516.

[Sub61] B. A. Subbotovskaja. "Realization of linear functions by formulas using $\vee$, &, $-$". In: *Soviet Math. Dokl.* 2 (1961), pp. 110–112. ISSN: 0197-6788. URL: http://mi.mathnet.ru/eng/dan24539.

[SVW17] Thomas Steinke, Salil Vadhan, and Andrew Wan. "Pseudorandomness and Fourier-Growth Bounds for Width-3 Branching Programs". In: *Theory Comput.* 13 (2017), Paper No. 12. DOI: 10.4086/toc.2017.v013a012.

[ŠŽ11] Jiří Šíma and Stanislav Žák. "Almost k-Wise Independent Sets Establish Hitting Sets for Width-3 1-Branching Programs". In: *Proceedings of the 6th International Computer Science Symposium in Russia (CSR)*. 2011, pp. 120–133. DOI: 10.1007/978-3-642-20712-9_10.

[SZ95] Michael Saks and David Zuckerman. Unpublished. 1995.

[SZ99] Michael Saks and Shiyu Zhou. "$\mathrm{BP_HSPACE}(S) \subseteq \mathrm{DSPACE}(S^{3/2})$". In: *J. Comput. System Sci.* 58.2 (1999), pp. 376–403. ISSN: 0022-0000. DOI: 10.1006/jcss.1998.1616.

[Tal14] Avishay Tal. "Shrinkage of De Morgan Formulae by Spectral Techniques". In: *Proceedings of the 55th Symposium on Foundations of Computer Science (FOCS)*. 2014, pp. 551–560. DOI: 10.1109/FOCS.2014.65.

[Tal17] Avishay Tal. "Tight Bounds on the Fourier Spectrum of $\mathbf{AC^0}$". In: *Proceedings of the 32nd Computational Complexity Conference (CCC)*. 2017, 15:1–15:31. DOI: 10.4230/LIPIcs.CCC.2017.15.

[Tel18] Roei Tell. "Quantified Derandomization of Linear Threshold Circuits". In: *Proceedings of the 50th Symposium on Theory of Computing (STOC)*. 2018, pp. 855–865. DOI: 10.1145/3188745.3188822.

[Tom81] Martin Tompa. "An extension of Savitch's theorem to small space bounds". In: *Inform. Process. Lett.* 12.2 (1981), pp. 106–108. ISSN: 0020-0190. DOI: 10.1016/0020-0190(81)90013-2.

[TS18] Amnon Ta-Shma. *Space-Bounded Computation*. Course at Tel-Aviv University. 2018. URL: https://www.cs.tau.ac.il/~amnon/Classes/2018-Space/class.htm.

[TX13] Luca Trevisan and TongKe Xue. "A Derandomized Switching Lemma and an Improved Derandomization of AC0". In: *Proceedings of the 28th Conference on Computational Complexity (CCC)*. 2013, pp. 242–247. DOI: 10.1109/CCC.2013.32.

[Tzu09] Yoav Tzur. "Notions of Weak Pseudorandomness and $GF(2^n)$-Polynomials". M.Sc. thesis. Weizmann Institute of Science, 2009. URL: https://eccc.weizmann.ac.il/static/books/Notions_of_Weak_Pseudorandomness/.

[Uma03] Christopher Umans. "Pseudo-random generators for all hardnesses". In: *J. Comput. System Sci.* 67.2 (2003). Special Issue on STOC 2002, pp. 419–440. ISSN: 0022-0000. DOI: 10.1016/S0022-0000(03)00046-1.

[Uma20] Chris Umans. *Current Topics in Theoretical Computer Science*. Course at the California Institute of Technology. 2020. URL: https://web.archive.org/web/20200703104639/http://users.cms.caltech.edu/~umans/cs153/index.html.

[Vad12] Salil P. Vadhan. "Pseudorandomness". In: *Foundations and Trends® in Theoretical Computer Science* 7.1–3 (2012), pp. 1–336. ISSN: 1551-305X. DOI: 10.1561/0400000010.

[Vio07] Emanuele Viola. "Pseudorandom Bits for Constant-Depth Circuits with Few Arbitrary Symmetric Gates". In: *SIAM J. Comput.* 36.5 (2007), pp. 1387–1403. ISSN: 0097-5397. DOI: 10.1137/050640941.

[Vio14] Emanuele Viola. "Randomness Buys Depth for Approximate Counting". In: *Comput. Complexity* 23.3 (2014), pp. 479–508. ISSN: 1016-3328. DOI: 10.1007/s00037-013-0076-6.

[Vio15] Emanuele Viola. "The communication complexity of addition". In: *Combinatorica* 35.6 (2015), pp. 703–747. ISSN: 0209-9683. DOI: 10.1007/s00493-014-3078-3.

[Wat13] Thomas Watson. "Pseudorandom generators for combinatorial checkerboards". In: *Comput. Complexity* 22.4 (2013), pp. 727–769. ISSN: 1016-3328. DOI: 10.1007/s00037-012-0036-6.

[Yao82] Andrew C. Yao. "Theory and Applications of Trapdoor Functions". In: *Proceedings of the 23rd Symposium on Foundations of Computer Science (FOCS)*. 1982, pp. 80–91. DOI: 10.1109/SFCS.1982.45.

[Zuc18] David Zuckerman. *Simple Optimal Hitting Sets for Small-Success RL (oral presentation)*. Presented at "Princeton Theory Lunch". 2018. URL: https://www.youtube.com/watch?v=qq8tsU3BXb0.

[Zuc97] David Zuckerman. "Randomness-Optimal Oblivious Sampling". In: *Random Struct. Algorithms* 11.4 (Dec. 1997), pp. 345–367. ISSN: 1042-9832. DOI: 10.1002/(SICI)1098-2418(199712)11:4<345::AID-RSA4>3.0.CO;2-Z.

Vita

William M. Hoza was born in Olympia, Washington on June 2, 1994. After being raised by two wonderful and loving parents, he graduated from St. Michael Parish School's kindergarten in 2000. He received a B.S. in mathematics and computer science from the California Institute of Technology in 2016, and he started graduate school at the University of Texas at Austin later that year. He married his wife Alicia in 2019, and he currently has two brothers, two sisters, two sisters-in-law, and two brothers-in-law.

Email Address: whoza@utexas.edu

This dissertation was typeset with LaTeX 2_ε[1] by the author.

[1] LaTeX 2_ε is an extension of LaTeX. LaTeX is a collection of macros for TeX. TeX is a trademark of the American Mathematical Society. The macros used in formatting this dissertation were written by Dinesh Das, Department of Computer Sciences, The University of Texas at Austin, and extended by Bert Kay, James A. Bednar, and Ayman El-Khashab.